СПРАВОЧНИК

Б. В. ШЕВКОПЛЯС

МИКРО-ПРОЦЕССОРНЫЕ СТРУКТУРЫ

•

ИНЖЕНЕРНЫЕ РЕШЕНИЯ

2-е издание,
переработанное
и дополненное

МОСКВА „РАДИО И СВЯЗЬ"
1990

ББК 32.97
Ш37
УДК 681.325.5—181.4(035)

Рецензент канд. техн. наук В. А. Шахнов

Редакция литературы по электронике

Шевкопляс Б. В.

Ш37 Микропроцессорные структуры. Инженерные решения: Справочник. — 2-е изд. перераб. и доп. — М.: Радио и связь, 1990. — 512 с.: ил.

ISBN 5-256-00460-3.

Рассмотрены практические вопросы проектирования микропроцессорных устройств и систем на их основе. Приведены методы и схемы разделения и расширения адресного пространства, примеры построения последовательных каналов связи, даны практические рекомендации по обеспечению помехозащищенности устройств. Рассмотрено свыше двухсот оригинальных структурных и схемных решений. В отличие от первого издания (1986 г.) повышена информативность, описаны новые технические решения.

Для инженерно-технических работников, занимающихся вопросами проектирования, производства и эксплуатации средств вычислительной техники.

Ш $\frac{2302030700—173}{046(01)—90}$ 92-90 ББК 32.97

Справочное издание

ШЕВКОПЛЯС БОРИС ВЛАДИМИРОВИЧ

МИКРОПРОЦЕССОРНЫЕ СТРУКТУРЫ. ИНЖЕНЕРНЫЕ РЕШЕНИЯ

Справочник

Заведующий редакцией Ю. Н. Рысев. Редактор М. М. Лисина. Переплет художника Н. А. Пашуро. Художественный редактор Н. С. Шеин. Технический редактор И. Л. Ткаченко. Корректор З. Г. Галушкина

ИБ № 2101

Сдано в набор 27.03.90. Подписано в печать 30.07.90. Т-12613. Формат 84×108¹/₃₂. Бумага тип. № 2. Гарнитура Литературная. Печать высокая. Усл. печ. л. 26,88. Усл. кр.-отт. 26,88. Уч.-изд. л. 33,81. Тираж 100 000 экз. Изд. № 22882. Зак. № 528. Цена 2 р. 50 к.

Издательство «Радио и связь». 101000 Москва, Почтамт, а/я 693

Владимирская типография Госкомитета СССР по печати 600000, г. Владимир, Октябрьский проспект, д. 7

ISBN 5-256-00460-3

© Шевкопляс Б. В., 1990

Многие вещи нам непонятны не потому, что наши понятия слабы; но потому, что сии вещи не входят в круг наших понятий.

К. Прутков

Предисловие

В этой книге собраны описания ряда структурных и схемных решений, относящихся к микропроцессорной технике. Эти решения в большинстве своем являются изобретениями, т. е. отвечают критериям мировой новизны, существенных отличий и полезности. Поэтому данная книга, строго говоря, не является «справочником» в общепринятом смысле — это, скорее, собрание «жемчужин передовой схемотехнической мысли». Иными словами, это — «справочник идей», и в нем нет ни одной схемы с указанием конкретных типов элементов с их цоколевкой и т. п.

Тем не менее приведенная в книге информация вполне достаточна для воплощения схем в «железе» с учетом конкретных конструктивно-технологических ограничений, используемой элементной базы и т. п. Едва ли не каждое решение можно усовершенствовать, поэтому многие из них могут послужить аналогами и прототипами при создании новых изобретений.

Тем, кто мало знаком с микропроцессорной техникой, можно порекомендовать прочесть первый раздел книги, в котором приведены сведения, облегчающие понимание смысла решений, рассматриваемых во втором ее разделе.

Описания технических решений и рисунки даны в максимально упрощенном виде и, как правило, не совпадают с материалами первоисточников по форме. Что же касается содержания, то автор надеется, что привнесенные при упрощенном описании искажения (а они, к сожалению, неизбежны) не затрагивают сути решений.

В заключение хотелось бы пожелать читателю оптимизма и хорошего делового настроения, которое поможет оценить «красоту» и «мощь» некоторых приведенных здесь схем.

Раздел I

ВВЕДЕНИЕ В ТЕХНИКУ МИКРОЭВМ

Глава 1

Выбор микропроцессорной системы и организация адресного взаимодействия ее составных частей

1.1. С чего начать проектирование микропроцессорной системы?

С четкой формулировки задачи. Задача считается хорошо сформулированной, если ее решение можно представить в виде блок-схемы алгоритма концептуального уровня [1], как показано на примерах 1.1 и 1.2.

Пример 1.1. Работа с объектами в режиме циклического обслуживания (рис. 1.1).

Описание функций блоков:

1. Инициализировать систему при включении напряжения питания.
2. Проверить терминал № 1 и, если это необходимо, обменяться с ним информацией.
3. Проверить все остальные терминалы и произвести их обслуживание, если это необходимо.
4. Прочитать состояние всех входных информационных линий системы и произвести вычисления, если это необходимо.
5. Вычислить новые значения управляющих сигналов и передать их на выходные линии системы.
6. Произвести регенерацию очередной цифры 10-разрядного семисегментного индикатора.
7. Ждать 10 мс.

Из этого примера видно, что проработка задачи выполнена на уровне, исключающем «наукообразную» неопределенность, — каждый блок выполняет вполне определенную функцию, которую можно детализировать, не прибегая к догадкам и предположениям. Содержание задачи не представляет сейчас для нас интереса — важно уяснить степень детализации ее решения на данном уровне. Отметим только, что в системе используется широко распространенный метод программного последовательного опроса внешних источников информации. Значительная часть «интеллекта» микропроцессора, состав-

ляющего ядро системы, тратится на то, чтобы следить за сигналами внешнего мира.

Пример 1.2. Работа с объектами в режиме прерываиия (рис. 1.2).

Описание функций блоков:

1. Инициализация системы при включении напряжения питания.
2. Последовательная циклическая регенерация всех цифр светодиодного индикатора в период ожидания прерывания.
3. Переход к соответствующей программе обслуживания терминала при получении сигнала прерывания.
4. Ввод данных с терминала № 1.
5. Ввод данных с терминала № 2.
6. Ввод данных с терминала № n.

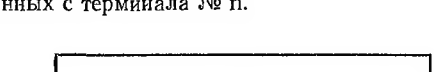


Рис. 1.1. Блок-схема алгоритма на уровне концепций (первый вариант)

7. Вычисление текущего значения результата.
8. Перекодировка результата для вывода на дисплей.

В данном случае после включения напряжения питания и приведения системы в исходное состояние (с участием или без участия человека), т. е. после инициализации системы, микропроцессор «зацикливается» в блоке 2 и, например, через каждые 10 мс поочередно подсвечивает цифры светодиодного индикатора, так что человеческий глаз не успевает заметить их мерцания.

Любое существенное для этой системы событие вызывает срабатывание схем, заставляющих микропроцессор временно прервать выполнение действий, предписанных блоком 2, и через блок 3 перейти к программе реакции на это событие. Такой метод реакции на события внешнего мира называется методом прерываний.

После рассмотрения задачи на уровне концепций можно спуститься на второй — алгоритмический — уровень. Заметим, что ни первый, ни второй уровень проектирования не требует привязки к конкретному микропроцессору. (С высоты этих уровней все микропроцессоры представляются одинаковыми.) В то время как на первом уровне решался вопрос *что делать*, на втором уровне максимально подробно должен быть решен вопрос *как делать*.

Цель алгоритмического уровня — раскрыть содержание блоков уровня концепций. Согласно [1], каждому блоку первого уровня могут соответствовать 20 или более блоков второго уровня, т. е. дистанция между этими уровнями обычно велика. Степень детализации задачи на алгоритмическом уровне должна быть такой, чтобы каждый блок соответствовал одной, двум или трем машинным командам.

Пример 1.3. Блок-схема алгоритмического уровня — раскрытие блока 4 в примере 1.1 (рис. 1.3).

Описание функций блоков:

1. Установить нулевой код в счетчике входных слов.
2. Установить в регистре X адрес ячейки, в которой хранится

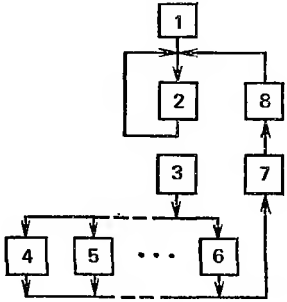


Рис. 1.2. Блок-схема алгоритма на уровне концепций (второй вариант)

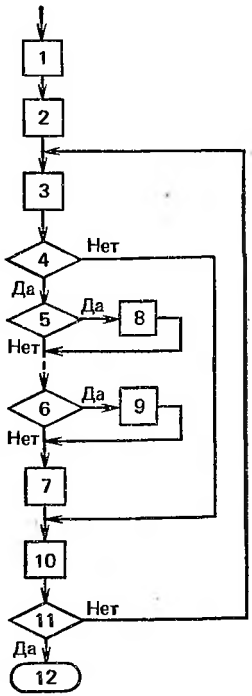


Рис. 1.3. Блок-схема алгоритмического уровня

старое значение входного слова (последующие ячейки соответствуют последующим словам).

3. Прочитать очередное входное слово и сравнить с предыдущим значением. Выделить несовпадающие позиции.

4. Отличия имеются?

5. Несовпадение в бите 0?

6. Несовпадение в бите N?

7. Запомнить новое входное слово в ячейке, отведенной для хранения предыдущего значения.

8. Выполнить действие.

9. Выполнить действие.

10. Прибавить единицу к содержимому счетчика слов и к содержимому регистра X.

11. Все слова обработаны?

12. Продолжение.

Алгоритм реализует чтение входных слов, каждое из которых состоит из N бит, и сравнение с ранее считанными значениями. Если имеются отличия, то это означает, что ситуация во внешнем мире изменилась и нужно предпринять соответствующие действия (блоки 8, 9).

После проработки задачи на алгоритмическом уровне определяются требования к микропроцессору, после чего выбирается конкретный микропроцессор. Система команд и архитектура выбранного микропроцессора являются основой для перехода от алгоритмического уровня к нижнему — командному уровню. Если система команд слишком примитивна, то, возможно, будет целесообразно ввести дополнительный, более подробный алгоритмический уровень.

Пример 1.4. Командный уровень — реализация блока 1 в примере 1.3:

```
CLR @ # S1.
```

Эта комаида предписывает микропроцессору установить в нуль содержимое ячейки памяти, которой соответствует символический адрес S1. Символ S1 выбран программистом произвольно (в пределах правил) для обозначеиия счетчика входных слов. Остальные символы этой записи представляют собой «жесткую заготовку», взятую из таблицы, описывающей систему команд конкретиой микроЭВМ («Электроника-60»). В данном случае блоку 1 примера 1.3 соответствует единствеиная команда.

Таким образом, проектирование микропроцессорной системы следует вести «сверху — вниз», чтобы на всех этапах ясно видеть задачу в целом.

1.2. Типовая структура микроЭВМ, принцип адресного взаимодействия ее составных частей

Типовая мнкроЭВМ (рис. 1.4) содержит объединеиные общей внутреиней магистралью процессор, память, коитроллеры и адаптер магистралей. Продолжением внутренией магистрали является внешняя магистраль, к которой в данном примере подключен контроллер,

Рис. 1.4. Функциоиальиая схема типовой микроЭВМ

связаиный с некоторым объектом управления, например с магнитным пускателем электродвигателя. В качестве устройств ввода-вывода могут, например, использоваться: графическнй дисплей, накопитель на гибких магнитных дисках и печатающее устройство.

Центральным звеном микроЭВМ является пара процессор — память, остальные элемеиты обеспечивают связь этого звеиа с внешним миром. Коитроллер предназначен для сопряжеиия одного или нескольких устройств ввода-вывода с внешней или виутреиией магистралью микроЭВМ. Магистраль реализуется в виде набора проводов илн печатных проводиików, по которым с помощью приемопередаюших элементов передаются различиого рода сигналы.

Адаптер магистралей выполияет фуикцию их электрического н, возможио, логического сопряжения. В простейшем случае он «прозрачен» с обеих сторои и выполняет лишь функцию электрического преобразования уровней сигналов, например маломощных сигналов

внутренней КМОП-магистрали с амплитудой 9 В и сигналов мощной внешней ТТЛ-магистрали с амплитудой 3,5 В.

Память обычно содержит постоянное и оперативное запоминающие устройства (ПЗУ и ОЗУ). Возможно использование перепрограммируемых постоянных запоминающих устройств (ППЗУ).

В памяти микроЭВМ имеется два вида информации — команды и данные, причем с «точки зрения» памяти эти виды информации неотличимы. О том, где находятся данные, а где — команды, «знает» только процессор. По образному выражению Э. Клингмана, «команды можно сравнить с глаголами естественного языка, указывающими на действие, которое должно быть выполнено, тогда как слова данных аналогичны существительным, являющимся либо подлежащим — предметом, совершающим действие, либо дополнением — предметом, на который переходит действие» [2].

МикроЭВМ может содержать таймер — устройство для отсчета времени. Это, по существу, обычные кварцевые часы, которые можно программно опрашивать, устанавливать, использовать в качестве будильника, в определенное время выдающего сигнал в процессор, или в качестве секундомера при замерах интервалов времени между внешними событиями и т. д. Точность хода таких часов может быть обычной, например ±2 с/сут.

В состав многих микроЭВМ входит так называемый сетевой таймер, который вырабатывает импульсы при прохождении питающего синусоидального напряжения 220 В, 50±1 Гц через «нуль» в одном из направлений. Эти импульсы, следующие с периодом 20 мс, поступают в процессор, и он при желании (программиста) может ими воспользоваться.

Какой таймер лучше: кварцевый или сетевой? Ответ на этот вопрос, казалось бы, очевиден — кварцевый. Действительно, кварцевые часы за сутки отстанут или убегут не более чем на 2 с, а «сетевые» могут ошибиться почти на полчаса. Поэтому, если иметь в виду только точностные характеристики и разрешающую способность, предпочтение следует отдать кварцевому таймеру. Однако эти характеристики не всегда являются определяющими. В некоторых применениях сетевой таймер оказывается вне конкуренции и даже самые точные кварцевые часы неспособны его заменить (см., например, § 5.2).

Процессор способен обмениваться информацией с объектами за его границами только в том случае, если эти объекты имеют адреса. Такие объекты называют адресуемыми или программно-доступными. Адрес — это код, однозначно определяющий тот или иной элемент внутри или вне микроЭВМ: ячейку памяти, регистр, триггер, вход логического элемента и т. п.

Элементарный цикл работы микроЭВМ состоит из двух этапов: процессор извлекает из памяти код, который он рассматривает как команду; затем он выполняет команду и готовится к выбору следующей команды. Такие циклы повторяются независимо от содержания выполняемой задачи. Из их последовательности складывается машинное время микроЭВМ. Поэтому можно утверждать, что процессор умеет выполнять только два действия: выбирать команды из памяти и выполнять их. (Говоря более строго, возможно и третье действие — стандартная реакция на внешние прерывания, но это действие будет рассмотрено особо.)

Список команд процессора насчитывает обычно порядка ста команд. В этот список, в частности, входят:

1. Безадресные комаиды, или команды типа регистр — регистр. При выполнении таких комаид действие производится иад содержимым одного или двух внутренних регистров процессора, результат также остается в одном из регистров. Например, комаида типа R5&R7→R5 предписывает процессору произвести логическое поразрядное умножение старого содержимого регистра R5 на содержимое регистра R7 и результат заиести в регистр R5. Выполнение этой комаиды не связано с обращением за «граиицы» процессора.

2. Адресиые комаиды, или команды типа регистр — память. Оии предписывают процессору производить действие с содержимым указанной в команде ячейки памяти. Например, команда типа [2007]∨ ∨R4→R4 предписывает процессору произвести логическую операцию поразрядной дизъюнкции содержимого ячейки памяти, адрес которой равен 2007, и старого содержимого внутреннего регистра R4 процессора с записью нового результата в регистр R4. Выполнение этой команды связано с извлечением из «внешнего мира» операнда для обработки. Важно, что под адресом 2007 может «скрываться» не ячейка памяти, а любой другой объект, который расценивает этот адрес как свой.

Функциональиая схема микроЭВМ, приведеиная на рис. 1.4, ие является единственно возможной. Можно представить себе более сложные структуры с несколькими виутрениими и внешними магистралими, с несколькими процессорами, с многовходовой памятью и т. д. Все эти структуры являются в большей или меньшей степени специализированными, т. е. отражающими особеиности типовой решаемой задачи. Здесь действует простой прииции: чем ближе аппаратная структура устройства к алгоритмической структуре решаемой задачи, тем лучше. Например, если задача допускает распараллеливание на три ветви со слабым взаимодействием между ветвями, то ее можио эффективно решать иа четырехпроцессориой системе, в которой одии из процессоров координирует работу трех остальных, не «мешающих» друг другу.

Одиако, прежде чем сделать первый шаг иа пути к специализации стандартных микропроцессорных устройств, иужно хорошо взвесить все *за* и *против*, поскольку это является крупным шагом иазад: ведь само появление в 70-х годах универсальиых микроЭВМ обусловлеио в первую очередь стремлением к унификации и удешевлеиию аппаратуры.

Построив специализированное микропроцессорное устройство, вы столкнетесь с рядом трудностей: с необходимостью разработки собственных программ, с проблемами модернизации, взанмозаменяемости и надежности и т. п. Поэтому после проработки задачи иа концептуальном или алгоритмическом уровне (см. § 1.1) важно не поддаться искушению разработать «собственное» микропроцессорное устройство. Многие поиачалу считают, что их задача уникальна и не решается без иагромождеиия специализированных блоков — всевозможиых одновибраторов, перекодировщиков, счетчиков и других «крайне необходимых» устройств. Другими словами, на этом этапе проектирования вы должны ответить на вопрос: иельзя ли использовать для решения задачи стандартиые микропроцессориые средства?

Передача адреса по магистрали. В примере, приведеином на рис. 1.5, магистраль типа «общая шина» [3] содержит 16 или 18 адресных линий — адресиую шину A, 16 информациоиных лииий —

шину данных D и до 22 линий управления — управляющую шину Control.

По адресной шине передаются адреса выбираемых устройств, по шине данных в ту или иную сторону (в зависимости от режима — чтение или запись) передаются данные, по шине управления — специальные сигналы (INITL — общий сброс, ACLO — авария сетевого питания, SSYN — ответ от ведомого и др.). В каждом акте обмена

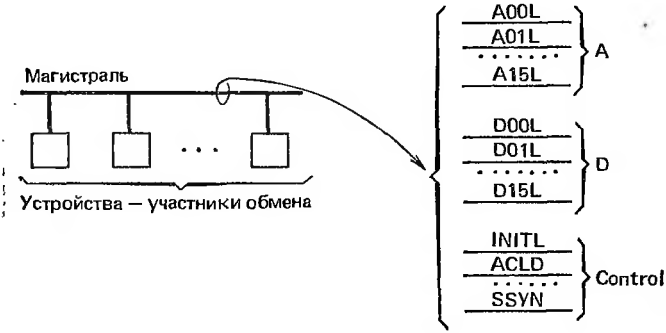


Рис. 1.5. Структура магистрали «общая шина» (стрелка указывает на развернутое изображение магистрали)

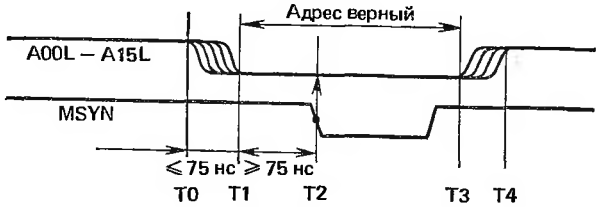


Рис. 1.6. Передача адреса по магистрали.

Активное устройство гарантирует истинность адреса во время действия сигнала MSYN

по магистрали участвуют только два устройства: активное (инициатор обмена, ведущее устройство) и пассивное (ответчик, ведомое устройство). Рассмотрим процесс передачи адреса от активного устройства к пассивному (рис. 1.6).

В момент времени T0 активное устройство начинает выдачу на все линии A00L — A15L напряжений низкого уровня. В данном примере выдается код 111...1, так как в магистрали «общая шина» напряжение низкого уровня (порядка 0,5 В) соответствует логической единице (лог. 1). С учетом разброса параметров передающих усилителей и других мешающих факторов сигналы на адресной шине примут верные значения только к моменту T1 (для магистрали ТТЛ

типа длиной 15 м, согласованной на концах, T1 — T0 ⩽ 75 нс). В момент времени T2 активное устройство формирует напряжение низкого уровня на линии MSYN синхронизации от ведущего. В этот момент существует гарантия того, что переходные процессы на линиях адресной шины закончились, как минимум, на 75 нс раньше. Эти 75 не могут использоваться для дешифрации адреса всеми пассивными устройствами. Отрицательный фронт сигнала MSYN может являться стробом опроса этих дешифраторов.

При получении ответного сигнала от единственного пассивного устройства, которое восприняло выданный адрес как свой, активное устройство снимает сигнал MSYN, а затем в промежутке времени T3 — T4 освобождает адресную шину. В пассивном состоянии на всех линиях адресной шины устанавливается напряжение высокого уровня (приблизительно 3,5 В).

Рассмотренный способ передачи адреса, конечно, не является единственно возможным. Активное устройство, например, может не заботиться о том, воспринят адрес одним из пассивных устройств или нет, если для передачи адреса отведен стандартный временной интервал во временной сетке, формируемой генератором импульсов. Все участники обмена при этом «знают», что в течение этого интервала адрес принял истинное значение. Это — один из возможных синхронных методов передачи адреса в отличие от предыдущего асинхронного.

Асинхронный метод позволяет обеспечить более высокую скорость обмена, так как при получении ответного сигнала можно сразу же освободить адресную шину. Что касается гарантированной задержки отрицательного фронта сигнала MSYN, то здесь существует следующее противоречие: с одной стороны, эта задержка должна быть достаточно большой для надежного восприятия адреса пассивным устройством, с другой — ее желательно уменьшить до возможного предела, чтобы ускорить передачу адреса. Приблизиться к этому пределу можно лишь на конкретном образце устройства путем проведения серии тщательных экспериментов, рискуя все же после этого получить ненадежную работу при «бесконечно малых» ухудшениях физических параметров магистрали, например в связи с добавлением новой печатной платы, при изменении температуры или вследствие каких-либо других причин.

На первый взгляд может показаться неразрешимой задача построения системы, в которой приемник, не дожидаясь сигнала сопровождения, сам «догадывается» о том, что адрес принял установившееся значение (действительно, откуда бы ему знать об этом?). Решение этой задачи приведено в § 7.9.

В дальнейшем будут использоваться общепринятые сокращенные формы записи двоичных кодов.

Восьмеричная запись: $1000101101111011 = 105573_8$, так как 000=0, 001=1, 010=2, 011=3, 100=4, 101=5, 110=6, 111=7.

Шестнадцатеричная запись: $1000101101111011 = 8B7B_{16}$, так как 0000=0, 0001=1, 0010=2, 0011=3, 0100=4, 0101=5, 0110=6, 0111=7, 1000=8, 1001=9, 1010=A, 1011=B, 1100=C, 1101=D, 1110=E, 1111=F.

Адресное пространство. Если адресная шина содержит всего три сигнальные линии, то по ней можно передать $2^3 = 8$ различных адресов: 000, 001, 010, 011, 100, 101, 110, 111. При четырех линиях число возможных состояний шины составляет $2^4 = 16$: 0000, 0001, ...,

..., 1111; при 16 линиях число возможных состояний составляет $2^{16}=65536=64К$, где $К=2^{10}=1024$.

Адресное пространство представляет собой упорядоченное множество кодов 0, 1, 2, ..., (2^n-1); где n — число адресных линий. Это множество для наглядности представляют в виде отрезка числовой оси либо в виде таблицы, как показано на рис. 1.7 (n=16). Нумерация точек, или «ячеек», адресного пространства при его графическом представлении производится снизу вверх или сверху вниз (что удобнее) в десятичной, восьмеричной или шестнадцатеричной форме.

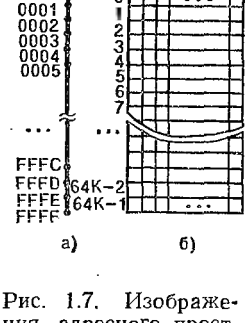


Рис. 1.7. Изображения адресного пространства:

а — в виде отрезка числовой оси; *б* — в виде таблицы с числом столбцов m, равным разрядности шины данных

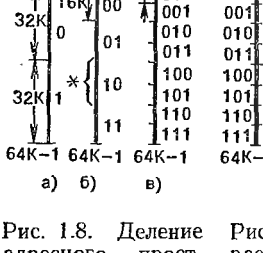


Рис. 1.8. Деление адресного пространства:

а — одним старшим разрядом адреса на две части; *б* — двумя старшими разрядами адреса на четыре части; *в* — тремя старшими разрядами адреса на восемь частей

Рис. 1.9. Пример распределения адресного пространства микроЭВМ:

А — зона 16К (0000—3FFF), отведенная для адресов ОЗУ; Б — зона 8К (4000—5FFF), отведенная для адресов ПЗУ; В — свободная зона 24К (6000—BFFF); Г — «окно» размером 8К (C000—DFFF) для адресации расширенной памяти; Д — зона 8К (E000—FFFF), отведенная для адресов регистров

Адресное пространство определяет число возможных отличимых друг от друга кодовых комбинаций (адресов), которые может выдать на адресную шину активное устройство. Это, конечно, не означает, что каждой такой комбинации соответствует программно-доступный элемент. Адресное пространство определяет лишь потенциальные возможности системы — максимальное число программно-доступных элементов, которые могут присутствовать в ней. Поэтому адресное пространство можно сравнить с пустым или частично заполненным помещением библиотеки, в котором можно разместить, например, не более 64К книг.

Старший двоичный разряд адреса делит адресное пространство

на две равные части (рис. 1.8, *а*), два старших разряда делят его на четыре равные части (рис. 1.8, *б*), три старших разряда — на восемь равных частей (рис. 1.8, *в*) и т. д. Шестнадцать разрядов адреса делят адресное пространство емкостью 64К на 64К частей по одной ячейке.

Предположим, что, подключив двухлучевой осциллограф к двум старшим разрядам адресной шины, мы в некоторый момент обнаружили на них кодовую комбинацию 10_2. Это означает, что в данный момент процессор обращается к некоторому программно-доступному элементу, лежащему в третьей четверти (считая от единицы, сверху) адресного пространства. Эта зона на рис. 1.8 помечена знаком * . Для уточнения позиции адресуемого процессором элемента необходимо было бы анализировать один или несколько более младших разрядов и в предельном случае для абсолютно точного определения «цели» необходимо анализировать все без исключения оставшиеся разряды.

Понятие адресного пространства позволяет наглядно представлять размещение в нем различных программно-доступных объектов. Пример распределения адресного пространства микроЭВМ показан на рис. 1.9. В адресном пространстве размещены адреса, принадлежащие ОЗУ и ПЗУ. Последние 8К ячеек отведены для адресации регистров контроллеров, входящих в микроЭВМ, и, возможно, регистров самого процессора. Эта зона в основном свободна. Зона C000 — DFFF отведена для расширения адресного пространства методом окна, который будет рассмотрен в § 1.5. Зона 6000 — BFFF не занята.

Если процессор обратится в любую незанятую ячейку адресного пространства, то он не получит ответного сигнала (в «общей шине» — сигнала SSYN) и «зависнет». Чтобы состояние зависания не длилось слишком долго, в процессоре обычно предусматривается схема формирования искусственного ответа, которая по истечении, например, 10 мкс принудительно завершает цикл обмена по магистрали и заставляет процессор перейти к специальной программе реакции на зависание. В ходе выполнения этой программы оператору может быть выдано сообщение в виде печатного текста либо выполнены любые другие действия, определяемые программой.

1.3. Методы дешифрации адресов

Предположим, что в нашем распоряжении имеется микроЭВМ с распределением адресного пространства, показанным на рис. 1.9, и мы хотим подключить к внешней магистрали этой микроЭВМ некоторый программно-доступный элемент, например D-триггер. Для определенности предположим, что внешняя магистраль (см. рис. 1.4, 1.5) микроЭВМ выполнена в виде «общей шины».

Для решения задачи подключения триггера нужно, во-первых, определить его «положение» в адресном пространстве и, во-вторых, предусмотреть схему адресного дешифратора, который срабатывает при обращении процессора к триггеру. Положение адреса триггера в адресном пространстве микроЭВМ можно выбрать в любой свободной области, т. е. либо в оставшейся незанятой части зоны E000 — FFFF, либо в области 6000 — BFFF. Остановимся, например, на последней: 6000 — BFFF. Структура дешифратора зависит от

используемой элементной базы и, что особенно важно, от выбранного метода дешифрации.

Задание адреса с использованием полной дешифрации. Предположим, что адрес, соответствующий триггеру, выбран равным BFFF, т. е. задан с точностью до отдельной ячейки адресного пространства. Схема связи триггера с магистралью для этого случая показана на рис. 1.10.

Загрузка программно-доступного триггера D4 может производиться при выполнении процессором команды пересылки типа регистр — память: R2→[BFFF]. В данном случае предполагается, что

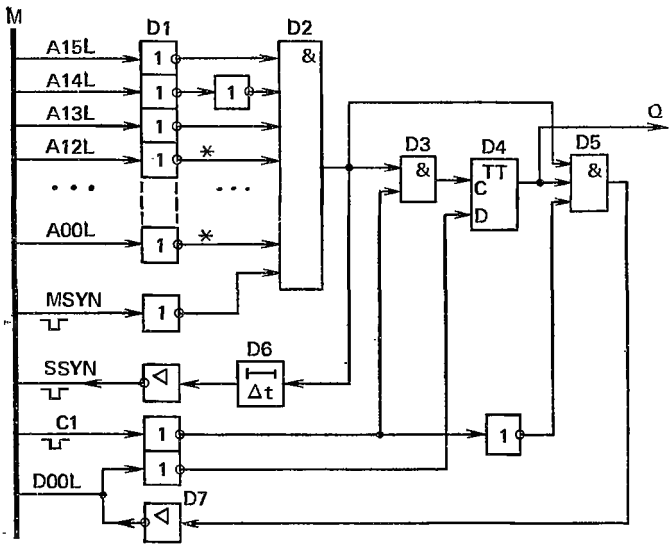


Рис. 1.10. Пример схемы полной дешифрации адреса.

Все сделано «строго по правилам», но, как часто бывает в таких случаях, можно сделать проще

в младший разряд внутреннего регистра R2 процессора предварительно занесен нужный бит (0 или 1), который при выполнении этой команды записывается в триггер D4. При выполнении указанной команды адресный код BFFF, выданный процессором в магистраль M, проходит через магистральные приемники D1 и далее вместе со стробирующим сигналом MSYN поступает на элемент D2, вызывая формирование единичного сигнала (напряжения высокого уровня) на его выходе. В режиме записи срабатывает элемент D3 и по положительному фронту сигнала на входе C триггер D4 принимает информацию с входа D.

Выдача информации из триггера D4 в линию D00L магистрали M производится при выполнении процессором команды чтения: [BFFF]→R2. В результате выполнения этой команды в младший

разряд регистра R2 процессора поступает информационный бит из триггера D4. Если триггер D4 находился в единичном состоянии, то при чтении срабатывает элемент D5 и последовательно соединенный с ним магистральный усилитель-инвертор D7, так что в линии D00L формируется напряжение низкого уровня («магистральная» лог. 1). Если триггер D4 находился в нулевом состоянии, то при чтении элемент D5 не срабатывает и в линии D00L за счет нагрузочных резисторов (на рисунке не показаны) поддерживается напряжение высокого уровня («магистральный» лог. 0).

При обращении к триггеру D4 по записи или по чтению в магистраль M выдается ответный сигнал SSYN. Элемент задержки D6 гарантирует выдачу сигнала SSYN не раньше, чем будет выдана информация из триггера D4 в линию D00L при чтении. (В предельном случае допустима одновременная выдача.)

Для обмена информацией между процессором и триггером D4 не обязательно использовать команды пересылки; можно воспользоваться, например, командой сложения или иной, в которой указана «ячейка памяти» с адресом BFFF. (В действительности ни в ОЗУ, ни в ПЗУ такой ячейки памяти нет, роль этой ячейки выполняет введенный нами адресуемый элемент — триггер D4.)

Задание адреса с использованием частичной дешифрации. Этот способ позволяет экономить аппаратуру дешифрации в результате присвоения адресуемому элементу (в данном случае — триггеру) не отдельной ячейки адресного пространства, а группы ячеек. Обратимся вновь к рис. 1.10 и будем считать, что связи, помеченные знаком *, устранены из схемы. В этом случае магистральные приемники D1 транслируют лишь три старших разряда адреса (вместо 16), а число входов элемента D2 сокращается с 17 до 4. Адреса, на которые реагирует дешифратор в этом случае, образуют массив размером в 8К ячеек: A000 — BFFF. Действительно, для обращения к триггеру D4 в этом случае необходимо в трех старших разрядах адреса создать комбинацию 101_2; при этом остальные 13 разрядов адреса могут быть заданы произвольно, одним из $2^{13}=8К$ способов. Другими словами, адрес триггера D4 можно представить так:

101 — — — — — — — — — — — — —,

где знак «—» означает, что данный разряд адреса не поступает на дешифратор и, следовательно, может быть задан произвольно.

Экономия аппаратуры в данном примере достигнута за счет того, что для одного адресуемого элемента (триггера) в адресном пространстве микроЭВМ пришлось выделить 8К ячеек из свободной зоны 6000 — BFFF. Напомним, что адресное пространство — абстрактное понятие и его стоимость, например, нельзя выразить в денежных единицах. Аппаратура дешифрации — понятие реальное, характеризуемое стоимостью, потребляемой мощностью, массой и т. д. Из всего этого вытекает следующее правило. Если вы обнаружили в адресном пространстве незанятую область, захватывайте для своих нужд как можно бо́льшую часть этой области, оставив, возможно, «на всякий случай» на будущее некоторую зону; тем самым вы сможете сэкономить аппаратуру дешифрации.

Передача информации к адресуемому элементу без использования шины данных. Пример построения схемы приведен на рис. 1.11. Задача та же, что и в двух предыдущих примерах (см. рис. 1.10): нужно обеспечить возможность программного управления состоянием выхода Q внешнего (пользовательского) триггера. Схема, показан-

ная на рис. 1.11, построена с использованием метода частичной дешифрации (она реагирует только на три старших разряда адреса), что, однако, не является принципиальным. Важнее то, что эта схема не подключена к шине данных и также не реагирует на управляющий сигнал записи C1 (см. рис. 1.10). Это дало возможность сэкономить два внешних контакта схемы. В схеме по существу имеются две программно-доступные точки — входы R и S триггера, которым присвоены следующие адреса:

адрес входа R: 101 — — — — — — — — — — — — — — — —;

адрес входа S: 100 — — — — — — — — — — — — — — — —.

Для установки триггера в единичное состояние программист предусматривает в программе адресную команду, например, такого типа: [8ABC]&R3→R3. При ее выполнении процессор пытается прочитать содержимое ячейки памяти с адресом 8ABC. Ячейки памяти с таким адресом в микроЭВМ нет (см. рис. 1.9). Вместо ответа от

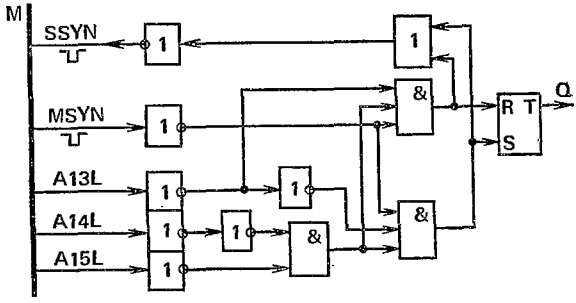


Рис. 1.11. Схема управления программно-доступным элементом (триггером) без использования шины данных

блока памяти в линию ответа SSYN будет выдан сигнал от нашей схемы — она реагирует на адрес 8ABC выдачей единичного сигнала на вход S триггера и посылкой сигнала SSYN в магистраль M (рис. 1.11). Процессор при этом «не подозревает о подлоге»: считывает с шины данных нулевой код (шина данных не подключена к нашей схеме), логически поразрядно умножает его на содержимое регистра R3 и результат записывает в регистр R3. Программиста, конечно, не интересует этот результат, так как указанная команда использовалась не по прямому назначению, а только для того, чтобы на адресной шине «промелькнул» код 8ABC, содержащий в трех старших разрядах комбинацию 100, соответствующую установке триггера в единичное состояние (Q=1).

Аналогично выполняется и сброс триггера. Для этого можно использовать, например, команду арифметического сложения содержимого внутреннего регистра R0 процессора с содержимым ячейки памяти с адресом BBBB (такой ячейки памяти нет): R0+[BBBB]→→R0.

В рассмотренном примере (рис. 1.11) информация о желаемом

состоянии триггера передается не по шине данных, а по адресной шине. Этот прием передачи информации «по умолчанию» известен еще из старых детективных историй, когда, например, факт телефонного звонка (адресного обращения) в назначенное время несет полезную информацию, хотя трубка не снимается и речевая информация не передается.

Из приведенных примеров можно сделать следующие выводы:

1. Понятие адреса позволяет присваивать имена самым различным объектам физического мира: триггерам, регистрам, клавишам пульта управления, мощным пусковым устройствам электроавтоматики и т. д.

2. Обезличивая таким образом физическую природу этих объектов, мы делаем их неотличимыми (с формальной точки зрения) от ячеек памяти, для работы с которыми выделен класс адресных команд. Используя такие команды и соответствующие аппаратные дешифраторы, мы имеем возможность программного доступа к этим объектам, поэтому они и называются программно-доступными.

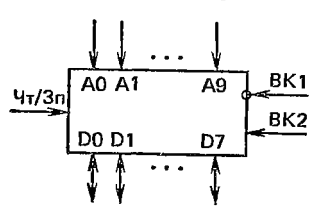


Рис. 1.12. Микросхема ОЗУ (1К×8 бит)

3. Процессор способен в нужной последовательности, определяемой программой, создавать импульсы в любых точках вашей аппаратуры, если этим точкам присвоены адреса.

4. Для передачи информации можно использовать или не использовать шину данных.

Примеры построения микроЭВМ с использованием принципа частичной дешифрации. *Пример 1.5.* Пусть в нашем распоряжении имеются три микросхемы: процессор и две микросхемы ОЗУ со структурой 1К×8 бит (в каждой микросхеме ОЗУ имеются 1024 8-разрядные ячейки). Как построить из них микроЭВМ с минимальными затратами дополнительной аппаратуры?

Рассмотрим сначала микросхему ОЗУ (рис. 1.12). При выполнении условий «выбора кристалла» ВК1=0 и ВК2=1, линии D0 — D7 работают как входы или выходы в зависимости от сигнала на линии чтение — запись (Чт/Зп). При остальных сочетаниях сигналов на выводах ВК1 и ВК2 (00, 10, 11) входы — выходы D0 — D7 отключены от внешней шины данных и находятся в высокоомном состоянии.

Простейшая схема подключения микросхем памяти к процессору может выглядеть так, как показано на рис. 1.13 [4].

Для правильной работы такой микроЭВМ необходимо, чтобы информация не выдавалась в шину данных одновременно из микросхем ОЗУ DS1 и DS2. Это условие выполняется, так как сигналы ВК1 и ВК1* никогда не принимают одновременно нулевых значений. При A15=0 работает микросхема DS1, а при A15=1 — микросхема DS2 (принята положительная логика).

Линии A0 — A9 адресной шины определяют адрес ячейки памяти внутри выбранной для обмена микросхемы DS1 или DS2. Линии A10 — A14 не используются. Линия чтение — запись не показана, как и другие несущественные для данного примера линии.

На рис. 1.14 показано распределение адресного пространства

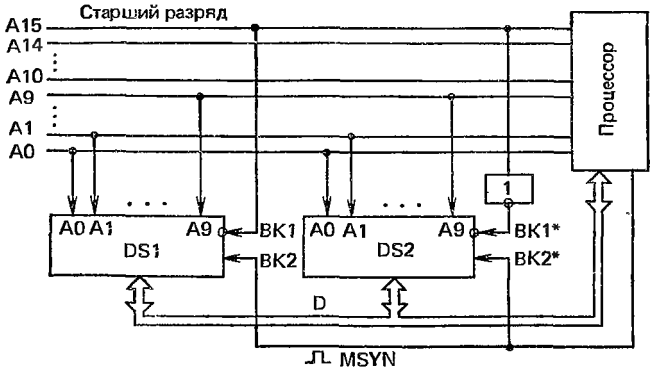


Рис. 1.13. МикроЭВМ с частичной дешифрацией адресов

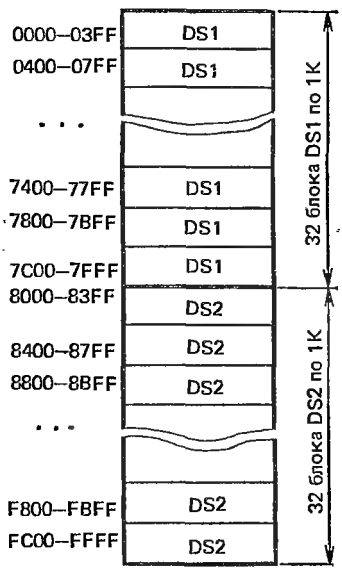


Рис. 1.14. Распределение адресного пространства микроЭВМ с частичной дешифрацией адресов; для выбора кристалла использована линия A15

микроЭВМ. Все адресное пространство оказалось заполненным «изображениями» микросхем DS1 и DS2. Действительно, по какому бы адресу ни обратился процессор, он обязательно попадает в одну из микросхем DS1 или DS2, поскольку в старшем разряде адреса присутствует нуль или единица (третьего не дано). Младшие десять разрядов определяют положение ячейки в одной из зон DS1 или DS2. Разряды A10—A15 определяют положение блока в адресном пространстве.

Поскольку все адресное пространство заполнено, этот пример имеет, скорее, иллюстративное, чем практическое значение. Он показывает возможности метода частичной дешифрации применительно к нашей задаче: из дополнительной аппаратуры использован лишь один инвертор.

Если для управления входами выбора кристалла микросхем DS1 и DS2 использовать, например, адресную линию 14-го, а не 15-го разряда, то распределение адресного пространства окажется иным (рис. 1.15, *а*). Однако и в этом случае все адресное пространство оказывается заполненным «изображениями» микросхем DS1 и DS2.

То же наблюдается и при использовании линий A13 — A10. На рис. 1.15, *б* показано распределение адресного пространства при использовании для выбора микросхемы линии A10.

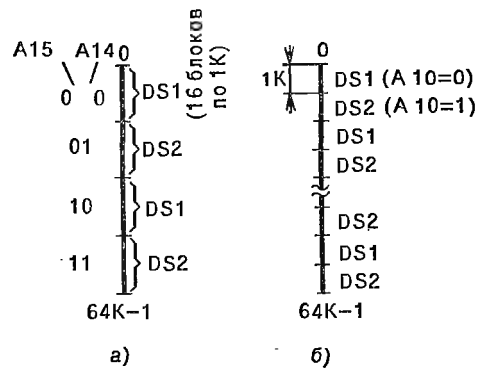


Рис. 1.15. Распределение адресного пространства микроЭВМ с частичной дешифрацией адресов; для выбора кристалла использована линия A14 *(а)* и линия A10 *(б)*

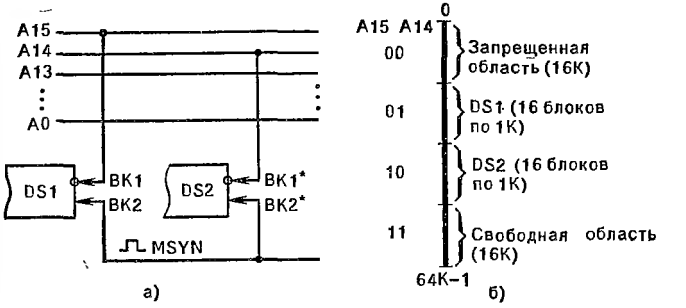


Рис. 1.16. Вариант подключения входов выбора кристалла микросхем ОЗУ к адресной шине *(а)* и соответствующее ему распределение адресного пространства *(б)*

Если исключить из схемы инвертор, вход ВК1 микросхемы DS1 подключить к линии A15, а вход ВК1* микросхемы DS2 — к линии A14 (рис. 1.16, *а*), то получим распределение адресного пространства, приведенное на рис. 1.16, *б*.

Первые 16К адресов (область 00) соответствуют одновременному выбору микросхем DS1 и DS2. При чтении одновременный выбор этих микросхем недопустим. Это связано с тем, что возможна

одновременная выдача «противофазной» информации (например, нулей из микросхемы DS1 и единиц из микросхемы DS2) в общую шину данных. Такая выдача приводит к электрической перегрузке выходных каскадов микросхем ОЗУ, и они могут выйти из строя. Однако режим записи в область 00 допустим (перегрузок не возникает). При этом одна и та же записываемая информация попадает в одноименные ячейки микросхем DS1 и DS2.

Это свойство данной схемы может быть использовано, например, для повышения достоверности информации за счет ее дублирования.

Если в программе предусмотрено сохранение в памяти переменной X, то эта переменная записывается по некоторому адресу, принадлежащему «запрещенной» (для чтения) области 00 (рис. 1.16, *б*). В дальнейшем, когда эта переменная должна быть извлечена из памяти, производится последовательное считывание двух равноценных «копий» этой переменной из ячеек, принадлежащих областям 01 и 10. Например, если процессор записывает код X в ячейку 3CAE, то этот код попадает одновременно в две ячейки с адресами 0AE, размещенные внутри микросхем DS1 и DS2. Считывание «копий» кода X ведется, например, из ячеек 70AE и B8AE. Если коды, считанные из обеих ячеек, одинаковы, то они совпадают с «оригиналом» и одна из «копий» может использоваться для вычислений. Если коды в указанных ячейках разные, то это означает, что один из них ошибочен, какой именно — неизвестно (предполагается, что ошибка локализована внутри одной из микросхем DS1 или DS2). При обнаружении несравнения процессор прекращает работу по основной программе и переходит к подпрограмме реакции на ошибку.

Область 11 (рис. 1.16, *б*) свободна и может использоваться для размещения в ней дополнительных адресуемых элементов.

Если ввести в систему третью микросхему ОЗУ (DS3) и подключить ее вход выбора кристалла ВК** к линии A13 (рис. 1.17, *а*), то получим распределение адресного пространства, показанное на рис. 1.17, *б*. Как и в предыдущей схеме, запись в «запрещенную» область 000 приводит к «размножению» информации. Одиако на этот раз получаем три одинаковые копии, доступные по чтению из областей 011, 101 и 110.

Рабочая программа, как и в предыдущем примере, предусматривает сохранение переменной X в области 000 и считывание двух копий этой переменной, например, из областей 011 и 110. Если копии совпадают, то одна из них используется для дальнейших вычислений. Если копии разные, то процессор игнорирует их и считывает «истинную» третью копию из области 101. (Принято предположение о неисправности, искажающей информацию только в одной БИС ОЗУ.) Третья копия в этом случае должна совпадать с первой или второй.

Отметим, что микросхемы DS1, DS2 и DS3 могут рассматриваться как независимые ОЗУ, размещенные в областях 011, 101 и 110 с общей емкостью 3К байт (а не 1К байт в резервированном варианте использования). Эти микросхемы можно использовать попарно, обеспечивая дублирование информации, как было показано ранее (см. рис. 1.16). При этом запись информации осуществляется в «запрещенные» области 001, 010 или 100 (рис. 1.17). Примечательно, что перестройка архитектуры микроЭВМ (переход от резервированной к нерезервированной структуре) производится чисто про-

граммным путем, без использования какой-либо внешней аппаратуры.

Пример 1.6. Структура микроЭВМ МЕК 6800 D1 [4], использующая частичную дешифрацию адресов (рис. 1.18).

МикроЭВМ содержит шесть однотипных микросхем ОЗУ (ОЗУ1 — ОЗУ6), микросхему ПЗУ, два адаптера магистралей АМ1 и АМ2 (две одинаковые микросхемы) и микросхему последовательного интерфейса ПИ. Функции, выполняемые процессором и памятью, уже кратко рассматривались в § 1.2. О микросхемах АМ1, АМ2 и ПИ пока достаточно знать лишь то, что они предназначены для связи этой микроЭВМ с внешним миром.

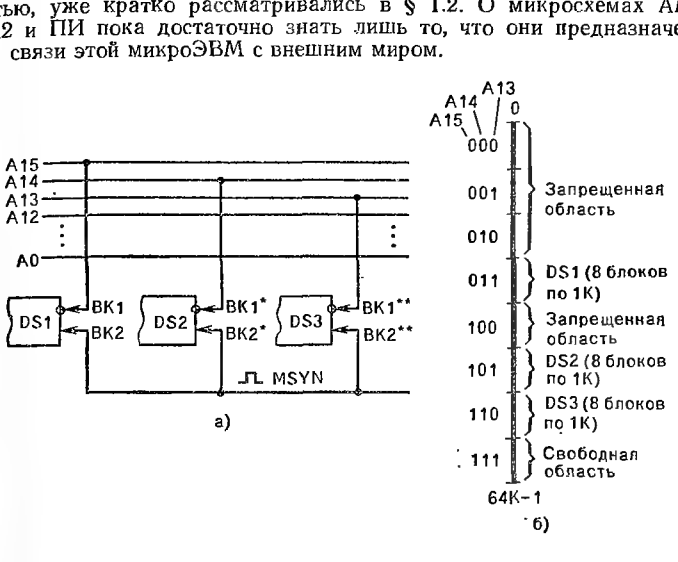


Рис. 1.17. Система с частичной дешифрацией, позволяющая обнаруживать и исправлять одиночные ошибки ОЗУ (*а*); распределение адресного пространства системы (*б*)

На рис. 1.18 показана только адресная шина. Шина данных и шина управления подключаются ко всем устройствам параллельно. Информация об использовании адресных линий в компактной форме сведена в табл. 1.1.

Каждое устройство содержит адресные входы А и группу входов выбора кристалла, выделенную на рис. 1.18 фигурной скобкой.

Микросхема ОЗУ содержит 128 8-разрядных ячеек, которые адресуются семью адресными линиями А0 — А6 ($2^7=128$). Каждая микросхема ОЗУ имеет четыре инверсных и два прямых входа выбора кристалла. Для того чтобы микросхема включилась в процесс записи или считывания, необходимо на все четыре инверсных входа ВК подать лог. 0, а на оба прямых входа ВК — лог. 1 (принята положительная логика). Неиспользуемые инверсные входы ВК должны быть подключены к шине нулевого потенциала («схемной земле»). Неиспользуемые прямые входы ВК всех микросхем объединяются в группы, которые через резисторы соединяются с шиной положи-

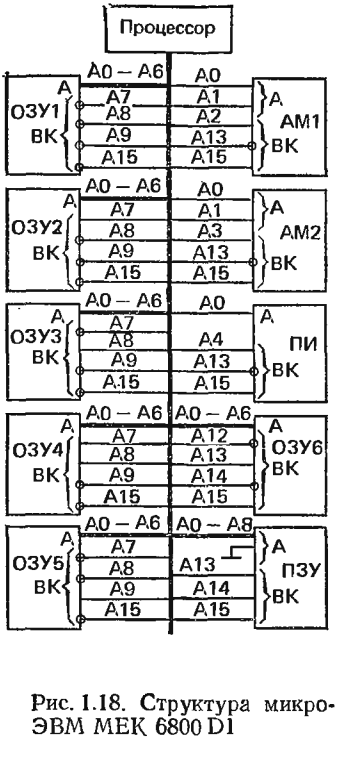


Рис. 1.18. Структура микроЭВМ МЕК 6800 D1

тельного источника питания. Из рис. 1.18 и табл. 1.1 видно, что для выбора ячейки внутри, например, ОЗУ4 нужно послать в линии 15 и 9 адресной шины лог. 0 (напряжения низкого уровня), а в линии 7 и 8 — лог. 1. При этом младшие семь разрядов адреса A0—A6 указывают адрес ячейки в пределах микросхемы ОЗУ4, а в разрядах A10—A14 может быть установлен произвольный код.

Мнкросхема ПЗУ имеет четыре входа ВК, из которых использованы три. Хотя ПЗУ содержит 1024 ячейки, по «историческим причинам» используется лишь 512 ячеек — девятый разряд адреса ПЗУ отключен от адресной шины микроЭВМ и заземлен (см. рис. 1.18). В табл. 1.1 этот факт отражен символом * в последней строке. (Этот символ можно было бы заменить знаком «—», так как разряд A9 не участвует в выборе микросхемы ПЗУ.)

Микросхемы АМ1, АМ2 и ПИ содержат по одному инверсному и по два прямых входа ВК. Адаптеры АМ1 и АМ2 содержат по четыре внутренних регистра, ПИ — два внутрениих регистра. Нас сейчас не интересуют функции, выполняемые этими микросхемами. Будем рассматривать их как программно-доступные со стороны процессора объекты.

Поскольку ни одно из устройств, кроме процессора, не использует для адресации все линии адресной шины, каждое из них занимает в адресном пространстве зону, бо́льшую, чем та, которая ему необходима.

Рассмотрим, например, ОЗУ6. Эта микросхема имеет емкость 128 ячеек, однако она поглощает область адресного пространства размером 4К ячеек, так как разряды с 7-го по 11-й не используются для адресации. Действительно, каждая ячейка ОЗУ6 может быть выбрана одним из $2^5=32$ способов, по числу комбинаций нулей и единиц на пяти иеиспользуемых линиях адресной шины. Поскольку число ячеек виутри микросхемы равно 128 и к каждой из них можно обратиться 32 способами, то область адресного пространства, покрываемая микросхемой ОЗУ6, равна $32\times128=4096=4К$.

Аналогично размножаются по 32 раза области ОЗУ1 — ОЗУ5. Область ПЗУ повторяетси в адресном пространстве восемь раз, причем 50 % этой областн оказываются недоступными для процессора за счет заземления старшего разряда непосредствеино на микросхеме (см. рис. 1.18).

Рассмотрим устройства АМ1, АМ2 и ПИ. Если A15=1, A13=0,

A2=1, то выбирается АМ1. Если A15=1, A13=0, A3=1, то выбирается АМ2. Однако если A15=1, A13=0, A2=1, A3=1, то выбираются оба устройства одновременно, что (при чтении) недопустимо. Более того, при A15=1, A13=0, A4=1, A3=1, A2=1 выбираются одиовременно АМ1, АМ2 и ПИ. Если программное обеспечение микроЭВМ отлажено до такой степени, что указанные запрещенные адреса не встречаются в программах, то описанная ситуация не возникает. Однако принципиальная возможность создания электрической перегрузки БИС по инициативе «умного программиста» напоминает бомбу замедленного действия.

Чтобы гарантировать невозможность одиовременного выбора двух или трех устройств (АМ1, АМ2, ПИ), можно ввести в микроЭВМ дополнительные дешифраторы (три элемента ИЛИ), как показано на рис. 1.19. С учетом этой доработки строки табл. 1.1, соответствующие устройствам АМ1, АМ2 и ПИ, будут выглядеть так, как показано в табл. 1.2.

Адреса устройств в данном случае не пересекаются, каждое из них занимает в адресном пространстве область размером 2К ячеек, суммарная занятая область составляет 2К×3=6К ячеек.

Вернемся к схеме, показанной на рис. 1.18. Каждое из устройств АМ1, АМ2 и ПИ занимает в адресном пространстве область размером 8К ячеек. Действительно, устройство АМ1 содержит четыре адресуемых элемента, однако их «изображения» повторяются в адресном пространстве 2^{11}=2К раз, так как 11 разрядов адресной шины не участвуют в дешифрации. То же относится и к устройству АМ2. Устройство ПИ содержит два адресуемых элемента, но их «изображения» повторяются в адресном пространстве 2^{12}=4К раз.

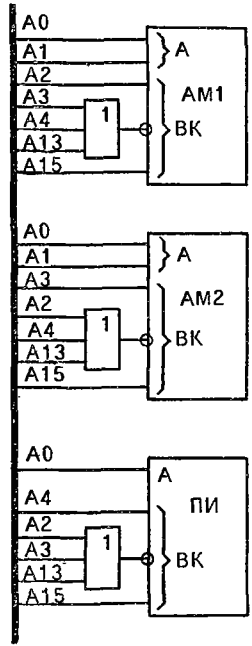


Рис. 1.19. Подключение к схеме на рис. 1.18 элементов ИЛИ для предотвращения одновременного выбора двух или трех устройств

Устройства АМ1, АМ2 и ПИ, вместе взятые, занимают в адресном пространстве область размером 14К (а не 8К×3=24К) из-за того, что адреса этих устройств частично пересекаются. Области 100 и 110 (рис. 1.20), отведенные для адресации этих устройств, можно представить состоящими из 256 зон по 32 ячейки с однотипным заполнением, причем в каждой такой зоне имеются 4 незанятые ячейки из 32 (1/8 часть). Таким образом, «плотность» адресов, на которые «откликаются» устройства АМ1, АМ2 или ПИ, составляет 7/8, и это справедливо для обеих областей 100 и 110. Поэтому среди 16К адресов, принадлежащих областям 100 и 110, имеется 16К×7/8=14К занятых и 2К незанятых адресов (последние «разбросаны» по четверкам с периодом, равным 32).

Таблица 1.1

Устройство	Разряды адреса															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ОЗУ1	0	—	—	—	—	—	0	0	0	×	×	×	×	×	×	×
ОЗУ2	0	—	—	—	—	—	0	0	1	×	×	×	×	×	×	×
ОЗУ3	0	—	—	—	—	—	0	1	0	×	×	×	×	×	×	×
ОЗУ4	0	—	—	—	—	—	0	1	1	×	×	×	×	×	×	×
ОЗУ5	0	—	—	—	—	—	1	0	0	×	×	×	×	×	×	×
АМ1	1	—	0	—	—	—	—	—	—	—	—	—	—	1	×	×
АМ2	1	—	0	—	—	—	—	—	—	—	—	—	1	—	×	×
ПИ	1	—	0	—	—	—	—	—	—	—	—	1	—	—	—	×
ОЗУ6	1	0	1	0	—	—	—	—	—	×	×	×	×	×	×	×
ПЗУ	1	1	1	—	—	—	*	×	×	×	×	×	×	×	×	×

Примечания. 1. Знак «—» означает, что данный разряд адреса не используется. 2. Знак × означает 0 или 1. Группа знаков ××...× определяет число адресных входов устройства, так что если обозначить число знаков × в группе через k, то информационная емкость устройства составит 2^k ячеек.

Таблица 1.2

Устройство	Разряды адреса															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ОЗУ1	0	—	—	—	—	—	0	0	0	×	×	×	×	×	×	×
.	.		.		.		.		.		.		.		.	
АМ1	1	—	0	—	—	—	—	—	—	—	—	0	0	1	×	×
АМ2	1	—	0	—	—	—	—	—	—	—	—	0	1	0	×	×
ПИ	1	—	0	—	—	—	—	—	—	—	—	1	0	0	—	×
.	.		.		.		.		.		.		.		.	
ПЗУ	1	1	1	—	—	—	*	×	×	×	×	×	×	×	×	×
ОЗУ7	0	—	—	—	—	—	1	0	1	×	×	×	×	×	×	×

Несмотря на значительное «засорение» адресного пространства микроЭВМ ненужными копиями, в нем остались незанятые области. Для их выявления проведем последовательное разбиение адресного пространства на две, четыре, восемь и т. д. частей.

1. Разбиение на две части по 32К ячеек. Предположение о существовании свободной области размером 32К ячеек (первой или второй половины адресного пространства) не подтверждается, так как в той и другой имеются адресуемые элементы, например, ОЗУ1 и ПЗУ.

2. Разбиение на четыре части по 16К ячеек. Свободных областей такого размера нет, так как комбинациям двух старших разрядов

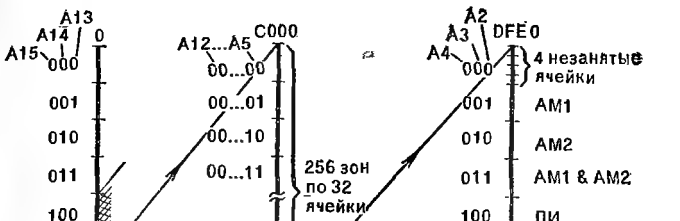


Рис. 1.20. Распределение адресного пространства в областях 100 и 110, отведенных для адресации устройств АМ1, АМ2 и ПИ

адреса 00 и 01 соответствует, например, выбор ОЗУ1, а комбинациям 10 и 11 — выбор АМ1.

3. Разбиение на 8 частей по 8К ячеек. Комбинациям 000—011 соответствует выбор ОЗУ1, комбинациям 100 и 110 — выбор АМ1, комбинации 101 — выбор ОЗУ6 и комбинации 111 — выбор ПЗУ.

4. Разбиение на 16 частей по 4К ячеек. Единственная кодовая комбинация в четырех старших разрядах, на которую не «откликается» ни один из адресуемых элементов, такова: $1011 = B_{16}$. Таким образом, в адресном пространстве микроЭВМ имеется непрерывная свободная область размером 4К ячеек: B000 — BFFF. Точнее говоря, к этой области примыкают еще четыре ячейки: C000 — C003.

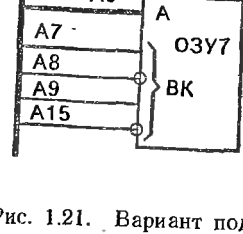


Рис. 1.21. Вариант подключения дополнительной микросхемы ОЗУ к адресной шине микроЭВМ (см. рис. 1.18)

Продолжая процесс разбиения адресного пространства, можно обнаружить и другие, менее протяженные свободные области. Если, например, нужно ввести в микроЭВМ еще одну микросхему ОЗУ (128×8 бит), то ее можно подключить, как показано на рис. 1.21, использовав одну из таких свободных областей. При этом в табл. 1.1 появится дополнительная строка, см. нижнюю строку табл. 1.2.

В рассмотренном примере построения микроЭВМ, благодаря применению принципа частичной дешифрации и удачному выбору схем подключения входов ВК микросхем к адресной шине, дополнительные схемы дешифрации отсутствуют. Однако занятое адресное пространство значительно величено по сравнению с минималь-

но необходимым для размещения в нем «единственных экземпляров» адресуемых элементов. Действительно, для размещения шести ОЗУ (128×8 бит) необходимо иметь $128 \times 6 = 768_{10}$ ячеек, ПЗУ — 512 ячеек, АМ1, АМ2 и ПИ — 10 ячеек, всего 1290 ячеек. В действительности шесть ОЗУ занимают область 4К×6=24К ячеек, АМ1, АМ2 и ПИ — 14К ячеек, ПЗУ — 8К ячеек, всего 46К ячеек. В результате кратность превышения затрат адресного пространства составляет $46 \times 1024/1290 \approx 36{,}51$.

1.4. Схемы адресных дешифраторов

Использование стандартных дешифраторов для разделения адресного пространства. На рис. 1.22 приведена схема дешифратора, имеющего три входа ВК выбора кристалла (a, b, c), три информационных входа (d, e, f) и восемь выходов Y0 — Y7. Если (a, b, c)≠≠(1, 0, 0), то Y0=Y1=...=Y7=1, т. е. дешифратор выключен. При (a, b, c)=(1, 0, 0), т. е. при обеспечении условий выбора кристалла, работа дешифратора описывается табл. 1.3.

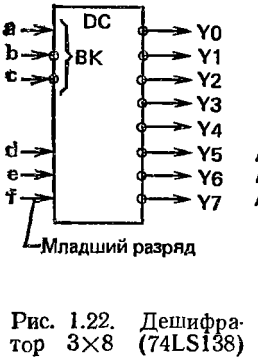


Рис. 1.22. Дешифратор 3×8 (74LS138)

Рис. 1.23. Разделение адресного пространства объемом 64К ячеек на восемь зон по 8К ячеек

Таблица 1.3

d	e	f	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
0	0	0	0	1	1	1	1	1	1	1
0	0	1	1	0	1	1	1	1	1	1
0	1	0	1	1	0	1	1	1	1	1
0	1	1	1	1	1	0	1	1	1	1
1	0	0	1	1	1	1	0	1	1	1
1	0	1	1	1	1	1	1	0	1	1
1	1	0	1	1	1	1	1	1	0	1
1	1	1	1	1	1	1	1	1	1	0

Рассмотрим несколько примеров использования этого дешифратора для разделения адресного пространства.

Пример 1.7. Подключим прямой вход ВК к линии сопровождения адреса MSYN, заземлив два других (инверсных) входа ВК. Информационные входы дешифратора подключим к трем старшим линиям адресной шины (рис. 1.23). Такое включение дешифратора позволяет разбить адресное пространство объемом 64К ячеек на восемь зон по 8К ячеек. Если подключить осциллограф, например, к выводу Y4 дешифратора, то, обнаружив отрицательный импульс на этом выводе, можно быть уверенным, что процессор в этот период времени обращался по одному из адресов, лежащих в зоне 8000—9FFF. Эта зона помечена на рис. 1.23 знаком *.

Пример 1.8. Схема включения, показанная на рис. 1.24, позволяет выделять в адресном пространстве восемь блоков по 2К ячеек.

Рис. 1.24. Выделение восьми блоков по 2К ячеек

Работа дешифратора возможна только тогда, когда на входы ВК поступит нужное сочетание сигналов, т. е. при A15=A14=0. Это произойдет при попадании адреса в первую четверть адресного пространства, вынесенную на рис. 1.24 для наглядности вправо с помощью стрелок m и n. В зависимости от сочетания нулей и единиц на входах A13—A11 мы попадаем в одну из восьми долей первой четверти адресного пространства, т. е. в зону размером 2К ячеек.

Пример 1.9. Схема последовательно включенных дешифраторов, приведенная на рис. 1.25, позволяет выделять восемь блоков по 256 ячеек.

Пример 1.10. Схема, приведенная на рис. 1.26, позволяет выделять восемь блоков по 16 ячеек.

Использование ПЗУ в качестве адресного дешифратора. Пусть требуется выделить зоны адресного пространства для размещения в них адресов следующих устройств, каждое из которых представляет собой одну микросхему:,

а) ПЗУ1 зону F000 — FFFF, 4К ячеек;
б) ПЗУ2 зону E000 — EFFF, 4К ячеек;
в) ОЗУ1 зону A000 — A7FF, 2К ячеек;
г) УВВ зону 8000 — 87FF, 2К ячеек;
д) ОЗУ2 зону 0000 — 3FFF, 16К ячеек.

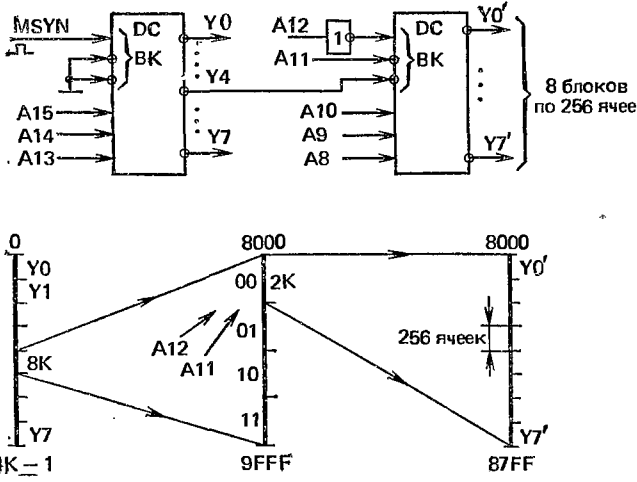


Рис. 1.25. Выделение восьми блоков по 256 ячеек

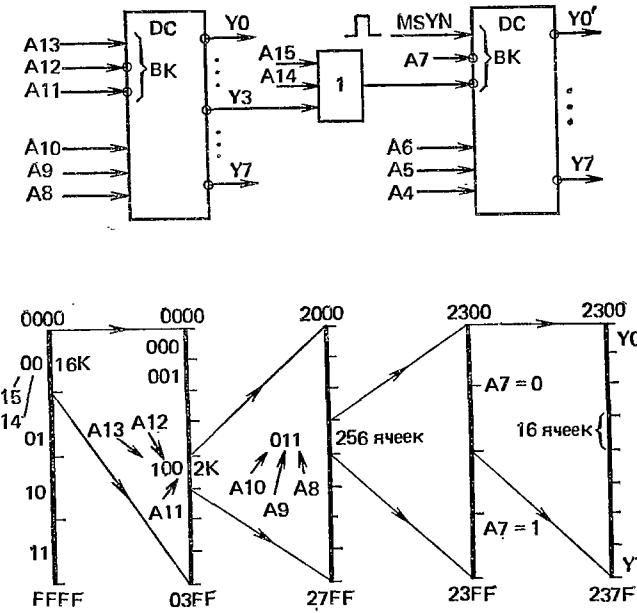


Рис. 1.26. Выделение восьми блоков по 16 ячеек

Так как наименьший блок имеет размер 2К ячеек, то разрешающая способиость дешифратора должна обеспечивать деление адресного пространства с точностью до зон размером 2К ячеек. Анализируя пять старших разрядов адреса, получаем необходимую точность, поскольку они делят все адресное пространство объемом 64К ячеек на $2^5=32$ части по 2К ячеек, что и требуется.

Выбираем за основу ПЗУ с пятью адресными входами, имеющее структуру 32×8 бит (рис. 1.27). Выходы А—Д этого ПЗУ подключаем к инверсным входам выбора кристалла соответствующих микросхем (см. условие задачи).

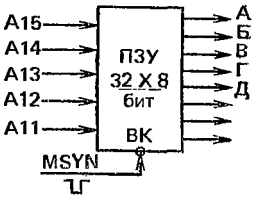


Рис. 1.27. Микросхема ПЗУ 32×8 бит, подключенная к адресной шине

Теперь задача сводится к тому, чтобы разметить все адресное пространство, продвигаясь по нему с шагом 2К ячеек и выбирая на каждом шаге желаемую выходную реакцию. Другими словами, задача сводится к разработке таблицы кодировки ПЗУ, которая в дальнейшем «зашивается» в микросхему. Содержимое ПЗУ представлено в табл. 1.4.

Если, например, старшие разряды адреса попадают в диапазон 00000 — 00111_2, то это означает, что полный 16-разрядный адрес лежит в диапазоне 0000—$3FFF_{16}$. При этом, согласно условию задачи, на выходе Д ПЗУ нужно закодировать 0 (см. табл. 1.4), что обеспечит выбор ОЗУ2. Программирование остальных условий производится аналогично.

Заметим, что в данном применении ПЗУ напоминает стандартный дешифратор, в котором нуль не «бежит по диагонали», как в табл. 1.3, а программируется по усмотрению пользователя (табл. 1.4). Перераспределение адресного пространства микроЭВМ достигается простой заменой микросхемы (кодировки ПЗУ). Для удобст-

Таблица 1.4

Диапазон адресов	Входы					Выходы				
	A15	A14	A13	A12	A11	А	Б	В	Г	Д
0000—07FF	0	0	0	0	0	1	1	1	1	0
0800—0FFF	0	0	0	0	1	1	1	1	1	0
1000—17FF	0	0	0	1	0	1	1	1	1	0
1800—1FFF	0	0	0	1	1	1	1	1	1	0
2000—27FF	0	0	1	0	0	1	1	1	1	0
2800—2FFF	0	0	1	0	1	1	1	1	1	0
3000—37FF	0	0	1	1	0	1	1	1	1	0
3800—3FFF	0	0	1	1	1	1	1	1	1	0
4000—47FF	0	1	0	0	0	1	1	1	1	1
. . . . .	.	.	.	.	.	.	.	.	.	.
E800—EFFF	1	1	1	0	1	1	0	1	1	1
F000—F7FF	1	1	1	1	0	0	1	1	1	1
F800—FFFF	1	1	1	1	1	0	1	1	1	1

ва замены микросхема ПЗУ может не припаиваться своими выводами к печатной плате, а устанавливаться на контактирующей колодке.

Недостаток использования ПЗУ в качестве дешифратора состоит в том, что с уменьшением кванта деления адресного пространства увеличивается число адресных входов ПЗУ. При этом объем накопительной матрицы растет пропорционально 2^n, где n — число входов ПЗУ. Поэтому применение ПЗУ оправдано при сравнительно грубом делении, т. е. при малом числе входов. Например, для выделения блока размером 64 ячейки ПЗУ должно иметь 10 адресных входов, что соответствует микросхеме со структурой 1К×8 бит — сравнительно сложной и дорогой.

Задача дешифрации усложняется, если выделяемая область «не вписывается» в сетку деления адресного пространства на 2, 4, 8 и т. д. частей. Предположим, что необходимо выделить область размером 16К ячеек для следующих трех случаев: область лежит в пределах 0000—3FFF; в пределах 0001—4000; в пределах 0002—4001.

В первом случае область лежит точно в первой четверти адресного пространства и для выявления факта попадания в нее необходимо анализировать два старших разряда адреса. При использовании ПЗУ в качестве адресного дешифратора его объем составил бы $2^2=4$ бита.

Во втором случае область смещена на одну ячейку. При этом дешифратор должен отличать кодовую комбинацию 0000 от 0001 и комбинацию 4000 от 4001 для выявления границ области. Поэтому необходимо анализировать все 16 разрядов адреса. Объем ПЗУ составляет $2^{16}=64$ К бит.

В третьем случае область смещена на две ячейки. Это несколько облегчает задачу так как младший разряд адреса не участвует в дешифрации. Поэтому объем ПЗУ составляет $2^{15}=32$К бит.

Использование программируемой логической матрицы в качестве адресного дешифратора. Программируемая логическая матрица (ПЛМ) представляет собой устройство, в которое можно «разместить» некоторую систему логических выражений, представленную в терминах булевой алгебры. На рис. 1.28 показана (условно) микросхема ПЛМ К556РТ1. Пример системы логических выражений, реализуемых этой ПЛМ:

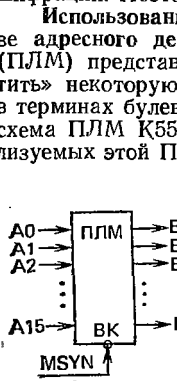


Рис. 1.28. Программируемая логическая матрица К556РТ1

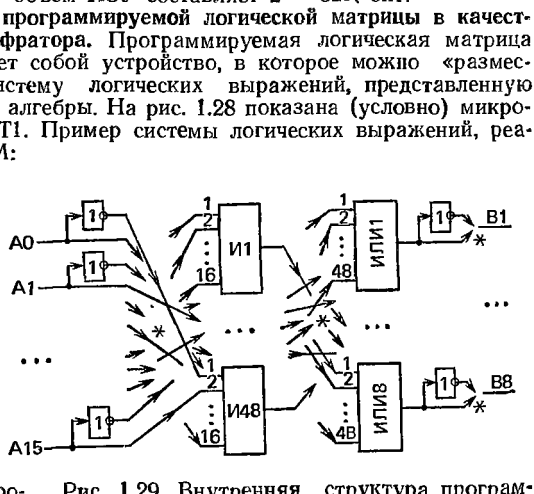


Рис. 1.29. Внутренняя структура программируемой логической матрицы

$$B1 = A1 \cdot \overline{A3} \cdot A5 \cdot A10 \vee \overline{A15 \cdot \overline{A14}} \vee \ldots \vee \ldots \vee \ldots;$$
$$\overline{B2} = \overline{A7 \cdot A12} \vee A1 \cdot A3 \cdot \overline{A5} \vee \ldots \vee \ldots \vee \ldots \vee \ldots;$$

$$B8 = \overline{A6 \cdot A8 \cdot \overline{A12} \cdot \overline{A14} \cdot A15} \vee \ldots \vee \ldots \vee \overline{A7 \cdot A12} \vee \ldots .$$

Записи, разделенные знаками дизъюнкции $\vee$, называются логическими произведениями. Число членов, разделенных знаками «·» (логическое умножение) в каждом произведении, не более 16. Общее число логических произведений не должно превышать 48. Это значение определяется площадью кристалла данной конкретной ПЛМ. Одинаковые логические произведения учитываются при подсчете общего числа только один раз (см. подчеркнутые логические произведения в приведеином выше примере). Любой выход может быть прямым или инверсным, по усмотрению пользователя. Выходы ПЛМ выполнены по схеме с «открытым коллектором».

Внутренняя структура ПЛМ представлена на рис. 1.29. До того как ПЛМ запрограммирована, имеются все связи, обозначенные символами *, т. е. для простоты можно считать, что каждый выход предыдущего уровня связан с каждым входом ближайшего последующего уровня. Ненужные связи устраняются на этапе программирования ПЛМ путем пережигания плавких перемычек.

Проведем сравнение ПЗУ и ПЛМ как адресных дешифраторов. Пусть в адресном пространстве требуется выделить следующие области:

D0 : 0000 — 3FFF (16К ячеек);
D1 : 8000 — 801F (32 ячейки);
D2 : A000—A07F (128 ячеек);
D3 : FE00 — FFFF (512 ячеек).

Эти же условия представлены в табл. 1.5.

Минимальной областью является область D1 размером в 32 ячейки. Для ее адресации с использованием ПЗУ необходимо подать на входы этого ПЗУ 11 старших разрядов адреса, с 15-го по 5-й. Поскольку $2^{11}=2048$, то необходимо ПЗУ со структурой $2К \times 4 = 8К$ бит (рис. 1.30). Теперь, используя информацию, представленную в табл. 1.5, записываем исходные условия в виде, удобном для их реализации в ПЛМ:

$$\overline{D0} = \overline{A15 \cdot \overline{A14}};$$
$$\overline{D1} = A15 \cdot \overline{A14} \cdot \overline{A13} \cdot \overline{A12} \ldots \overline{A5};$$
$$\overline{D2} = A15 \cdot \overline{A14} \cdot A13 \cdot \overline{A12} \cdot \overline{A11} \ldots \overline{A7};$$
$$\overline{D3} = A15 \cdot A14 \cdot A13 \ldots A9.$$

Эти формулы удовлетворяют перечисленным ограничениям и, следовательно, могут быть реализованы с помощью ПЛМ. Из 48 «име-

Таблица 1.5

Область	Разряды адреса															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D0	0	0	×	×	×	×	×	×	×	×	×	×	×	×	×	×
D1	1	0	0	0	0	0	0	0	0	0	0	×	×	×	×	×
D2	1	0	1	0	0	0	0	0	0	×	×	×	×	×	×	×
D3	1	1	1	1	1	1	1	×	×	×	×	×	×	×	×	×

ющихся» в ПЛМ логических произведений будут заняты только четыре. Дизъюнктивные возможности ПЛМ (...∨...∨...) в данном случае не используются. Если сравнивать ПЗУ и ПЛМ, то можно сказать, что в данном примере 8К бит ПЗУ эквивалентны четырем логическим произведениям, т. е. одно логическое произведение «стоит» 2К бит.

Поскольку каждое логическое произведение может содержать до 16 переменных, ПЛМ позволяет вести дешифрацию с точностью до отдельных ячеек, т. е. полную дешифрацию (не более чем восьми ячеек, расположенных в произвольных точках адресного пространства). Поскольку, как отмечалось, дизъюнктивные возможности ПЛМ не используются, специально для целей адресной дешифрации разработаны упрощенные ПЛМ без выходных элементов ИЛИ. Например, фирма Signetics, США выпускает ПЛМ 82S103 (16 входов, 9 выходов), PLS162 (16 входов, 5 выходов), PLS163 (12 входов, 9 выходов). Эти ПЛМ — оптимальные устройства для разделения адресного пространства.

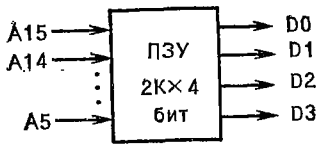


Рис. 1.30. Использование ПЗУ со структурой 2К×4 бит в качестве адресного дешифратора

Рассмотрим еще два примера использования ПЛМ для дешифрации адресных или иных кодов.

Пример 1.11. Предположим, что с помощью одной ПЛМ К556РТ1 необходимо разделить адресное пространство объемом 64К ячеек на три равные части с точностью плюс-минус одна ячейка. Эти части могут быть несплошными или сплошными.

1. Части несплошные. Возможное разбиение адресного пространства на части B0, B1 и B2 показано на рис. 1.31. Такому разбиению соответствуют следующие логические выражения (знаки «·» и × эквивалентны).

$$\overline{B0} = \overline{A15}\cdot\overline{A14} \vee A15\cdot A14\cdot\overline{A13}\cdot\overline{A12} \vee A15\cdot A14\cdot A13\cdot A12\cdot\overline{A11}\times$$
$$\times\overline{A10}\vee$$
$$A15\cdot A14...A10\cdot\overline{A9}\cdot\overline{A8}\vee A15\cdot A14...A8\cdot\overline{A7}\cdot\overline{A6}\vee$$

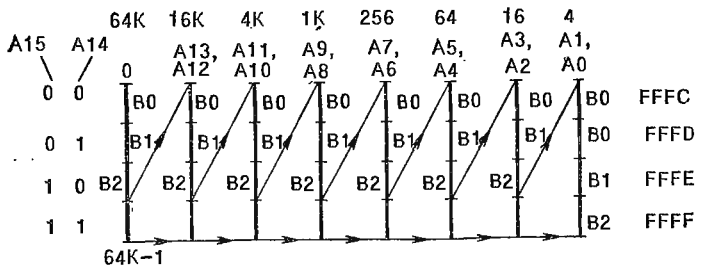


Рис. 1.31. Разделение адресного пространства на три несплошные части B0, B1 и B2

$$A15\cdot A14...A6\cdot\overline{A5}\cdot\overline{A4}\vee A15\cdot A14...A4\cdot\overline{A3}\cdot\overline{A2}\vee$$
$$A15\cdot A14...A2\cdot\overline{A1};$$

$$\overline{B1} = \overline{A15}\cdot A14 \vee A15\cdot A14\cdot\overline{A13}\cdot\overline{A12} \vee A15\cdot A14\cdot A13\cdot A12\cdot\overline{A11}\times$$
$$\times A10\vee A15\cdot A14...A10\cdot\overline{A9}\cdot A8\vee A15\cdot A14...A8\cdot\overline{A7}\cdot A6\vee$$
$$A15\cdot A14...A6\cdot\overline{A5}\cdot A4\vee A15\cdot A14...A4\cdot\overline{A3}\cdot A2\vee$$
$$A15\cdot A14...A1\cdot\overline{A0};$$

$$\overline{B2} = A15\cdot\overline{A14} \vee A15\cdot A14\cdot A13\cdot\overline{A12}\vee A15\cdot A14\cdot A13\cdot A12\cdot A11\times$$
$$\times\overline{A10}\vee$$
$$A15\cdot A14...A9\cdot\overline{A8}\vee A15\cdot A14...A7\cdot\overline{A6}\vee A15\cdot A14...A5\cdot\overline{A4}\vee$$
$$A15\cdot A14...A3\cdot\overline{A2}\vee A15\cdot A14...A0.$$

Если на вход ПЛМ поступает поток случайных кодов с равномерным распределением в диапазоне 0 — (64К—1), то с каждого из выходов B0—B2 снимается в три раза «прореженный» поток сигналов опознания соответствующей области.

2. Части сплошные. Разбиение адресного пространства поясняется табл. 1.6. Границы областей получены делением кода FFFF на три; F : 3=5, поэтому FFFF : 3=5555. Область B0 выбираем лежащей в диапазоне 0000—5554, область B1 — в диапазоне 5555—AAA9, область B2 — в диапазоне AAAA—FFFF. В соответствии с табл. 1.5:

$$\overline{B0} = \overline{A15}\cdot\overline{A14} \vee \overline{A15}\cdot A14\cdot\overline{A13}\cdot\overline{A12}\vee\overline{A15}\cdot A14\cdot\overline{A13}\cdot A12\cdot\overline{A11}\times$$
$$\times\overline{A10}\vee...\vee$$
$$\overline{A15}\cdot A14\cdot\overline{A13}\cdot A12\cdot\overline{A11}\cdot A10\cdot\overline{A9}\cdot A8\cdot\overline{A7}\cdot A6\cdot\overline{A5}\cdot A4\cdot\overline{A3}\times$$
$$\times A2\cdot\overline{A1}\cdot\overline{A0};$$

$$\overline{B1} = \overline{A15}\cdot A14\cdot\overline{A13}\cdot A12\cdot\overline{A11}\cdot A10\cdot\overline{A9}\cdot A8\cdot\overline{A7}\cdot A6\cdot\overline{A5}\cdot A4\cdot\overline{A3}\times$$
$$\times A2\cdot\overline{A1}\cdot A0\vee...\vee$$
$$A15\cdot\overline{A14}\cdot A13\cdot\overline{A12}\cdot A11\cdot\overline{A10}\cdot A9\cdot\overline{A8}\cdot A7\cdot\overline{A6}\cdot A5\cdot\overline{A4}\cdot A3\times$$
$$\times\overline{A2}\cdot\overline{A1};$$

$$\overline{B2} = A15\cdot\overline{A14}\cdot A13\cdot\overline{A12}\cdot A11\cdot\overline{A10}\cdot A9\cdot\overline{A8}\cdot A7\cdot\overline{A6}\cdot A5\cdot\overline{A4}\cdot A3\times$$
$$\times\overline{A2}\cdot A1\vee...\vee A15\cdot\overline{A14}\cdot A13\cdot A12\vee A15\cdot A14.$$

Пример 1.12. Определим кодировку ПЛМ, используемой в качестве адресного дешифратора, выделяющего область размером 16К ячеек для следующих трех случаев: область лежит в пределах 0000—3FFF; 0001—4000; 0002—4001.

1. Если область совпадает с первой четвертью адресного пространства (0000—3FFF), то для ее опознания достаточно одного логического произведения $\overline{B}=\overline{A15}\cdot\overline{A14}$. Это и есть искомая «кодировка».

2. Если область смещена на одну ячейку, то она описывается кодами, представленными в табл. 1.7.

Согласно табл. 1.7

$$\overline{B}=\overline{A15}\cdot\overline{A14}\cdot\overline{A13}...\overline{A1}\cdot A0\vee\overline{A15}\cdot\overline{A14}\cdot\overline{A13}...\overline{A2}\cdot A1\vee...\vee$$
$$A15\cdot\overline{A14}\cdot\overline{A13}\cdot A12\vee\overline{A15}\cdot\overline{A14}\cdot A13\vee\overline{A15}\cdot A14\cdot\overline{A13}\cdot\overline{A12}...\overline{A1}\cdot\overline{A0}.$$

Использовано 15 логических произведений.

2. Если область смещена на две ячейки, то она характеризуется кодами, представленными в табл. 1.8.

Таблица 1.6

Область	Разряды адреса															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
B0	0	0	×	×	×	×	×	×	×	×	×	×	×	×	×	×
	0	1	0	0	×	×	×	×	×	×	×	×	×	×	×	×
	0	1	0	1	0	0	×	×	×	×	×	×	×	×	×	×
	0	1	0	1	0	1	0	0	×	×	×	×	×	×	×	×
	0	1	0	1	0	1	0	1	0	0	×	×	×	×	×	×
	0	1	0	1	0	1	0	1	0	1	0	0	×	×	×	×
	0	1	0	1	0	1	0	1	0	1	0	1	0	0	×	×
	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	0
B1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
	0	1	0	1	0	1	0	1	0	1	0	1	0	1	1	×
	0	1	0	1	0	1	0	1	0	1	0	1	1	×	×	×
	0	1	0	1	0	1	0	1	0	1	1	×	×	×	×	×
	0	1	0	1	0	1	0	1	1	×	×	×	×	×	×	×
	0	1	0	1	0	1	1	×	×	×	×	×	×	×	×	×
	0	1	0	1	1	×	×	×	×	×	×	×	×	×	×	×
	0	1	1	×	×	×	×	×	×	×	×	×	×	×	×	×
	1	0	0	×	×	×	×	×	×	×	×	×	×	×	×	×
	1	0	1	0	0	×	×	×	×	×	×	×	×	×	×	×
	1	0	1	0	1	0	0	×	×	×	×	×	×	×	×	×
	1	0	1	0	1	0	1	0	0	×	×	×	×	×	×	×
	1	0	1	0	1	0	1	0	1	0	0	×	×	×	×	×
	1	0	1	0	1	0	1	0	1	0	1	0	0	×	×	×
	1	0	1	0	1	0	1	0	1	0	1	0	1	0	0	×
B2	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	×
	1	0	1	0	1	0	1	0	1	0	1	0	1	1	×	×
	1	0	1	0	1	0	1	0	1	0	1	1	×	×	×	×
	1	0	1	0	1	0	1	0	1	1	×	×	×	×	×	×
	1	0	1	0	1	0	1	1	×	×	×	×	×	×	×	×
	1	0	1	0	1	1	×	×	×	×	×	×	×	×	×	×
	1	0	1	1	×	×	×	×	×	×	×	×	×	×	×	×
	1	1	×	×	×	×	×	×	×	×	×	×	×	×	×	×

Согласно табл. 1.8

$$\overline{B} = \overline{A15}\cdot\overline{A14}\cdot\overline{A13}...\overline{A3}\cdot\overline{A2}\cdot A1 \vee \overline{A15}\cdot\overline{A14}\cdot\overline{A13}...\overline{A4}\cdot\overline{A3}\cdot A2 \vee ... \vee \overline{A15}\cdot\overline{A14}\cdot A13 \vee \overline{A15}\cdot A14\cdot\overline{A13}\cdot\overline{A12}...\overline{A2}\cdot\overline{A1}.$$

Использовано 14 логических произведений.

Для того чтобы в полной мере ощутить преимущества ПЛМ перед ПЗУ, рассмотрим пример, имеющий чисто «теоретическое» значение.

Пусть требуется реализовать на ПЗУ следующие функции: D0=A0; D1=A1; D2=A2; ...; D15=A15. Это функции проводов, соединяющих точки A_i с точками D_i (i=0, 1, 2, ..., 15), что поясняется рис. 1.32, *а*. Реализация этих функций на ПЗУ (рис. 1.32, *б*) требует

Таблица 1.7

Область	Разряды адреса															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
В	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	×
	0	0	0	0	0	0	0	0	0	0	0	0	0	1	×	×
	0	0	0	0	0	0	0	0	0	0	0	0	1	×	×	×
	0	0	0	0	0	0	0	0	0	0	0	1	×	×	×	×
	0	0	0	0	0	0	0	0	0	0	1	×	×	×	×	×
	0	0	0	0	0	0	0	0	0	1	×	×	×	×	×	×
	0	0	0	0	0	0	0	0	1	×	×	×	×	×	×	×
	0	0	0	0	0	0	0	1	×	×	×	×	×	×	×	×
	0	0	0	0	0	0	1	×	×	×	×	×	×	×	×	×
	0	0	0	0	0	1	×	×	×	×	×	×	×	×	×	×
	0	0	0	0	1	×	×	×	×	×	×	×	×	×	×	×
	0	0	0	1	×	×	×	×	×	×	×	×	×	×	×	×
	0	0	1	×	×	×	×	×	×	×	×	×	×	×	×	×
	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Таблица 1.8

Область	Разряды адреса															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
В	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	×
	0	0	0	0	0	0	0	0	0	0	0	0	0	1	×	×
	0	0	0	0	0	0	0	0	0	0	0	0	1	×	×	×
	0	0	0	0	0	0	0	0	0	0	0	1	×	×	×	×
	0	0	0	0	0	0	0	0	0	0	1	×	×	×	×	×
	0	0	0	0	0	0	0	0	0	1	×	×	×	×	×	×
	0	0	0	0	0	0	0	0	1	×	×	×	×	×	×	×
	0	0	0	0	0	0	0	1	×	×	×	×	×	×	×	×
	0	0	0	0	0	0	1	×	×	×	×	×	×	×	×	×
	0	0	0	0	0	1	×	×	×	×	×	×	×	×	×	×
	0	0	0	0	1	×	×	×	×	×	×	×	×	×	×	×
	0	0	0	1	×	×	×	×	×	×	×	×	×	×	×	×
	0	0	1	×	×	×	×	×	×	×	×	×	×	×	×	×
	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	×

накопителя, содержащего 64К×16=1М бит ($M=2^{20}$), так как в ответ на каждое входное сочетание переменных на выходы выдается «уникальный» код, равный в данном случае входному коду. Реализация этих же функций на ПЛМ требует шестиадцати «тривиальных» логических произведений: A0, A1, ..., A15. Физически это соответствует устранению всех «лишних» связей с сохранением тех, которые соединяют входы с соответствующими выходами (см. ис.

1.32, *а*). В данном примере одно логическое произведение соответствует 64К битам ПЗУ.

Не следует, однако, считать, что использование ПЛМ всегда более эффективно, чем использование ПЗУ. Если бы это было справедливо всегда, то ПЛМ полностью вытеснили бы ПЗУ, чего в действительности не происходит. Преимущества ПЛМ перед ПЗУ ощущаются при кодировании «осмысленной» (в той или иной степени упорядоченной) информации, в которой прослеживается регулярность. Например, в ПЛМ удачно «вписываются» микропрограммы, реализующие систему команд процессора, так как в этих микро-

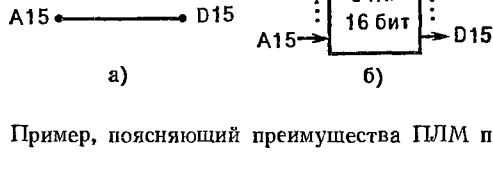


Рис. 1.32. Пример, поясняющий преимущества ПЛМ перед ПЗУ

программах много общих частей. «Случайная» информация, безусловно, лучше размещается в ПЗУ, чем в ПЛМ.

«В защиту» ПЗУ можно привести функцию, которая «не желает» укладываться в ПЛМ (составляющие ее логические произведения слишком чужеродны друг другу и не «склеиваются» между собой).

Для двух переменных это функция $f(2)=\overline{A0}\cdot\overline{A1}\vee A0\cdot A1$. Для трех переменных $f(3)=\overline{A2}\cdot\overline{A0}\cdot\overline{A1}\vee\overline{A2}\cdot A0\cdot A1\vee A2\cdot\overline{A0}\cdot A1\vee A2\cdot A0\cdot\overline{A1}$. Продолжая наращивать число переменных по закону $f(k+1)=A_k\cdot f(k)\vee A_k\cdot\overline{f(k)}$, получаем, что для 16 переменных число логических произведений составит 32К. Эта же функция может быть реализована с помощью ПЗУ объемом 64К×1 бит, т. е. одно логическое произведение эквивалентно двум битам ПЗУ. Учитывая, что в данном случае одно логическое произведение соответствует 16-входовому элемеиту И, а два бита ПЗУ реализуются на двух транзисторах, сравнение явно не в пользу ПЛМ (внутрениие дешифраторы ПЗУ для простоты не учитывались).

Следует отметить, что единого рецепта по выбору метода дешифрации и его воплощению в схемах не существует. Каждый из рассмотренных здесь по-своему хорош. Многое зависит от целей, которые вы преследуете, и от «рыночной конъюнктуры». Во всяком случае, полезио помиить обо всех рассмотренных вариантах.

После рассмотрения методов дешифрации понятие программно-доступного элемента обрело вполне конкретное «схемотехническое» содержание. Выбрав соответствующий задаче метод дешифрации и опираясь на рассмотренные примеры, можно построить схему, обеспечивающую такую доступность. Поэтому в дальиейшем термины «программно-доступный триггер» (регистр, блок и т. п.) будут использоваться без комментариев.

1.5. Методы расширения адресного пространства

Метод окна. При первом ознакомлении с задачей расширения адресного пространства многие предлагают ошибочный вариант решения, который в точности совпадает либо близок к приведенному на рис. 1.33, *а*. К шестнадцати основным адресным линиям добавлены еще четыре линии, которые подключены к выходам программно-доступного регистра старших разрядов адреса. Полный 20-разрядный адрес поступает в главную память емкостью $2^{20}=1$М ячеек. Шина данных, шина управления и схемы обеспечения программного доступа к регистру старших разрядов адреса на рисунке не показаны как несущественные (уже обсуждавшиеся ранее) элементы. Система оказывается неработоспособной, в чем можно убедиться, рассмотрев рис. 1.33, *б*.

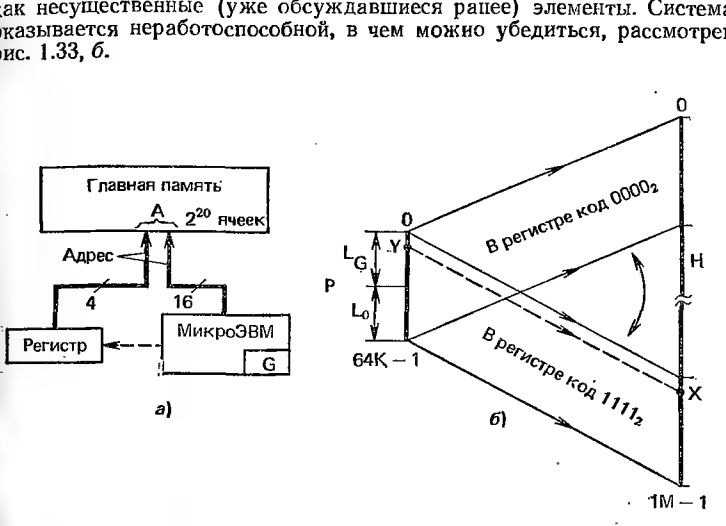


Рис. 1.33. Пример неправильного подключения главной памяти к микроЭВМ:

а — схема подключения; *б* — распределение адресного пространства; G — адресуемые элементы микроЭВМ — память, регистры и другие, занимающие не более 32К ячеек; P — адресное пространство микроЭВМ; H — адресное пространство главной памяти (16 зон по 64К ячеек); L_G — адресное пространство, занятое элементами G; L_0 — свободное адресное пространство микроЭВМ

Предположим, что из процессора микроЭВМ выдается адрес 0000000000000101_2, а в регистр старших разрядов адреса предварительно загружен код 1111_2. В главной памяти на этот код «откликнется» ячейка памяти X с адресом F0005. В то же время в собственной памяти микроЭВМ будет выбрана ячейка Y с адресом 0005, так как эта память анализирует 16, а не 20 разрядов адреса. (Собственная память микроЭВМ «не подозревает» о существовании главной памяти и связанной с ней дополнительной 4-разрядной адресной шины.)

Поэтому информация при чтении будет выдана как из собствен-

ной памяти микроЭВМ, так и из главной памяти, и на шине данных получится «смесь» сигналов. При записи код с шины данных попадет в обе ячейки одновременно, что также лишено смысла. Однако если исходный 16-разрядный адрес превышает или равен 32К, то одновременного обращения к двум ячейкам не будет, так как внутри микроЭВМ ни одно из устройств не опознает этот адрес, а в главной памяти ему с учетом четырех дополнительных старших разрядов соответствует единственная ячейка.

Этот пример представляет собой систему с частичной дешифрацией (см. § 1.3), где микроЭВМ является объектом, к которому подключены не все линии адресной шины системы (16 из 20). Поэтому адресное пространство микроЭВМ, как занятое, так и не занятое, размножено $2^4=16$ раз. В результате этого 50 % емкости главной памяти нельзя использовать. Вряд ли кто согласится работать с такой системой.

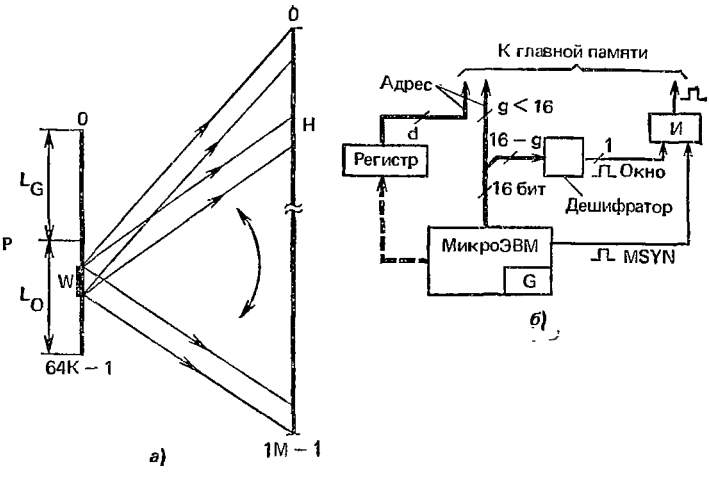


Рис. 1.34. Метод окна:

а — геометрическое представление идеи метода; *б* — схемная реализация; G — адресуемые элементы микроЭВМ, расположенные в зоне L_G адресного пространства P микроЭВМ; L_O — свободная зона адресного пространства, в которой выделено окно W; H — адресное пространство главной памяти

Метод окна позволяет очень просто разрешать указанные проблемы. Идея метода состоит в отображении на «большом» отрезке лишь части «малого» отрезка. Эта часть соответствует свободным адресам микроЭВМ и называется окном (рис. 1.34, *а*). Схемное решение приведено на рис. 1.34, *б*. Емкость главной памяти $V=2^{d+g}$ ячеек. Если, например, d=8, g=12, то $V=2^{20}=1$М ячеек. Главную память можно рассматривать как состоящую из 2^d страниц, каждая из которых имеет размер 2^g ячеек, совпадающий с размером окна. Таким образом, можно считать, что 2^d — число проекций окна, ко-

торые, примыкая вплотную друг к другу, заполняют собой всю «ось» главной памяти. Существенно, что адресная шина расщепляется на две части: одна поступает в главную память, а другая «поглощается» дешифратором. Система работает следующим образом.

1. Процессор загружает программно-доступный регистр старших разрядов адреса с шины данных, задавая положение проекции окна в адресном пространстве главной памяти или, в другой терминологии, угол наклона проецирующих лучей.

2. Процессор обращается по некоторому адресу, лежащему внутри окна.

3. Дешифратор опознает принадлежность текущего адреса фиксированному окну и разрешает передачу в главную память сигнала сопровождения адреса MSYN. В главную память выдается полный адрес (d+g бит), определяющий одну из ее ячеек.

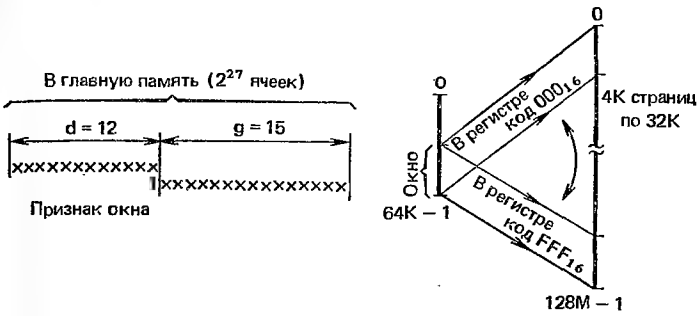


Рис. 1.35. Пример работы с окном в 32К ячеек

4. Выбранная ячейка главной памяти выдает или принимает информацию в зависимости от сигнала на линии управления режимом работы и посылает сигнал по линии передачи ответа (чтобы не загромождать рисунок, эти линии на рис. 1.34, *б* не показаны).

Поскольку окно выбрано в зоне неиспользуемых адресов внутреннего адресного пространства микроЭВМ, то конфликты, связанные с одновременным выбором двух адресуемых элементов (внутри и вне микроЭВМ), исключаются. Рассмотрим примеры задания окон.

Пример 1.13. Пусть имеется главная память емкостью $2^{27}=128$М ячеек. Ширина окна в адресном пространстве 16-разрядной микроЭВМ составляет 32К ячеек, а его позиция такова: 8000—FFFF. Требуется определить d и g.

Так как ширина окна составляет 32К ячеек, то из условия $2^g=32К=2^{15}$ получаем, что g=15; так как d+g=27, то d=27—15=12. Это решение поясняется диаграммами, приведенными на рис. 1.35. В главную память поступают разряды адреса, обозначенные символами ×. Признаком попадания в окно является единица в старшем разряде 16-разрядного адреса. Эта единица поглощается дешифратором (см. рис. 1.34, *б*) и не поступает в главную память, а служит лишь для «открывания ворот» к главной памяти.

Регистр старших разрядов адреса может загружаться в несколько этапов, если его разрядность превышает разрядность шины дан-

ных. Адрес регистра или адреса его составных частей должны, конечно, лежать вне окна. После загрузки регистра старших разрядов процессор как бы вставляет в зону 8000—FFFF своего адресного пространства одну из 4096 страниц емкостью 32К ячеек из главной памяти. Исчерпав информацию с этой страницы или, наоборот, сформировав нужный массив, процессор может вставить в свое адресное пространство любую другую страницу путем изменения содержимого регистра старших разрядов адреса. Заметим, что в данном примере дешифратор «вырождается» в провод, соединяющий его единственный вход с выходом.

Пример 1.14. Окно шириной 16К ячеек имеет позицию 8000—BFFF. Соответствующие диаграммы показаны на рис. 1.36.

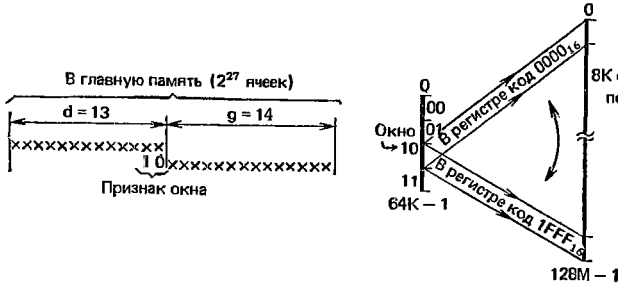


Рис. 1.36. Пример работы с окном в 16К ячеек

Чем шире окно, тем реже процессору приходится перезагружать регистр старших разрядов адреса (менять угол наклона проецирующих лучей) при передаче массива информации в ту или иную сторону. Поэтому для окна следует выделять по возможности бо́льшую свободную область адресного пространства.

Метод базовых регистров. Базовые регистры — это регистры старших разрядов. Метод базовых регистров является обобщением метода окна [5]. В адресном пространстве выделяется не одно, а несколько окон. При каждом обращении в главную память используется лишь одно окно (рис. 1.37). Проекции окон на «ось» главной памяти могут лежать в произвольных местах этой оси (с дискретом, равным размеру окна) и, в частности, могут совпадать. Положение окон в адресном пространстве задается жестко — с помощью дешифратора. Углы наклона лучей задаются содержимым базовых регистров. Размеры окон для простоты будем считать одинаковыми. Принцип формирования адресов главной памяти тот же, что и по методу окна.

Метод базовых регистров позволяет переносить в адресное пространство одновременно несколько различных страниц главной памяти, например одну страницу из области программ и две страницы из области данных. Схемная реализация метода базовых регистров близка к реализации метода окна и показана на рис. 1.38. Процессор загружает в базовые регистры с шины данных (на рисунке не показана) исходную информацию, определяющую углы наклона лучей из каждого окна. При обращении в главную память через одно

из m окон дешифратор формирует сигнал выдачи старших разрядов с соответствующего базового регистра через мультиплексор.

В рассмотренных примерах старшие разряды адреса из внешнего регистра «подстыковывались» к младшим, выдаваемым непосредственно с части 16-разрядной адресной шины. При такой «подстыковке» проекции окон в адресном пространстве главной памяти могут перемещаться с шагом, равным размеру окна. Для более плавного перемещения проекций вместо «подстыковки» используют аппаратное арифметическое суммирование содержимого базового регистра (его разрядность должна быть увеличена в сторону младших разрядов) с кодом, выдаваемым в адресную шину.

Старшие разряды кода, выдаваемого в 16-разрядную адресную шину, как и в приведенных примерах, «поглощаются» дешифратором и ие участвуют в формировании арифметической суммы.

В [13] предложена схема преобразования адресов, в которой 16-разрядный код из счетчика адреса команды процессора арифметически суммируется с содержимым 16-разрядного сегментного регистра, сдвинутым влево на n разрядов (n=0, 1, 2, ..., 16). Освободившиеся при сдвиге разряды заполняются нулями. Пусть, например, в счетчике адреса команды процессора записан код $022A_{16}$, в сегментном регистре — код $FB06_{16}$, n=8. После арифметического суммирования кодов 022A и FB0600 получим полный физический адрес, равный FB082A.

В зависимости от значения n разрядность полного физического адреса меняется от 17 до 32. Действительно, при n=0 максимальный (17-разрядный) код 1FFFE получается при суммировании двух 16-разрядных кодов, равных FFFF. При n=16 максимальный код FFFF из счетчика адреса команды процессора может суммироваться с кодом FFFF0000, в результате образуется 32-разрядный код FFFFFFFF, фактически полученный «подстыковкой» разрядов их обоих регистров.

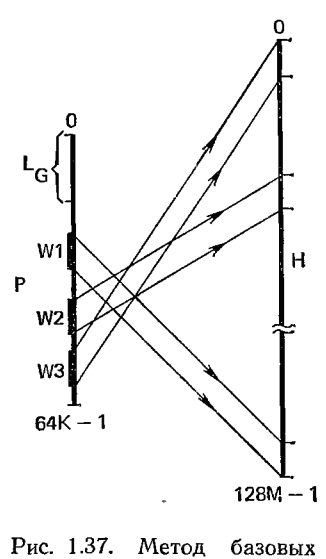


Рис. 1.37. Метод базовых регистров (осиовная идея):
P — адресное пространство микроЭВМ; L_G — занятая область; W1, W2, W3 — три окна; H — адресное пространство главной памяти

Увеличение разрядности полного физического адреса с увеличением n, однако, сопровождается уменьшением плавности перемещения проекций при отображении адресных пространств. Так, при n=0 суммирование затрагивает все разряды обоих кодов, в том числе и самые младшие. Поэтому, изменяя на единицу младшего разряда содержимое сегментного регистра, можно перемещать отображаемый массив с дискретностью в одну ячейку. При n=1 суммирование в самых младших разрядах не производится (четность

результата всегда совпадает с четностью кода в счетчике адреса команды), поэтому перемещение кода из счетчика адреса команды в расширенное адресное пространство возможно только по четным или нечетным адресам. Аналогично при n=2 дискретность перемещения составляет 4 ячейки, при n=3—8 ячеек и т. д.

Так как при работе процессора в составе той или иной микроЭВМ объем ОЗУ и ПЗУ может сильно меняться, в [13] предлагается задавать значение n программно, по результатам выявленной при включении напряжения питания конфигурации системы. В зависимости от имеющейся в распоряжении процессора физической па-

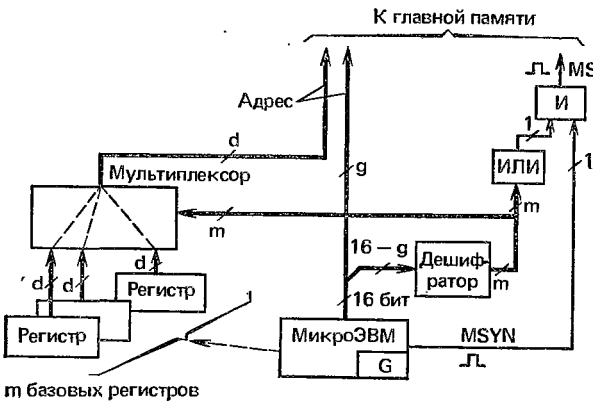


Рис. 1.38. Схемная реализация метода базовых регистров:

G — адресуемые элементы микроЭВМ, расположенные в зоне L_G ее адресного пространства

мяти, в специально выделенный внутренний регистр процессора заносится (и далее остается неизменным) оптимальное значение n, которое обеспечивает получение минимально необходимой разрядности физического адреса. Запас разрядности в данном случае вреден — он, как показано, привел бы к увеличению дискретности перемещения и, следовательно, к ненужным запасам при распределении физической памяти операционной системой.

Метод банков. Этот метод, как и оба предыдущие, получил достаточно широкое распространение [6—8, 199]. Под термином *банк* подразумевается отдельный блок памяти, содержащий, возможно, как оперативную, так и постоянную части. Помимо обычных магистральных входов-выходов такой блок памяти имеет дополнительный управляющий вход E разрешения работы. Схема подключения нескольких банков к микроЭВМ приведена на рис. 1.39.

В процессе работы такой системы процессор микроЭВМ через программно-доступный регистр открывает нужный банк и исполь-

зует информацию, которая в нем содержится. Остальные банки логически отключены от магистрали. В данном методе существенно то, что микроЭВМ может вообще не содержать собственной оперативной или постоянной памяти. Она может вставлять в свое адресное пространство емкостью, например, 64К банки емкостью 56К, оставляя последние 8К ячеек для собственных нужд, в частности для адресации показанного на рис. 1.39 регистра. Такая подстановка оказывается удобной для быстрого переключения микроЭВМ с решения одной задачи на решение другой. Каждой задаче в этом случае может соответствовать свой банк. При этом, однако, в каждом банке должны храниться одни и те же резидентные программы операционной системы, что снижает эффективность использования памяти в целом.

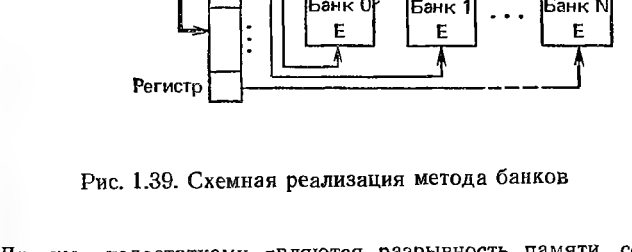


Рис. 1.39. Схемная реализация метода банков

Другими недостатками являются разрывность памяти, собранной из отдельных банков, невозможность одновременной работы с несколькими банками (с целью, например, быстрой переписи информации из одиого банка в другой), а также сложность доступа к этой памяти со стороны каналов прямого доступа в память, которые будут рассмотрены в гл. 2.

При включении напряжения питания регистр устанавливается в исходное состояние, соответствующее подключению к микроЭВМ единственного (вполне определенного) банка. Регистр может содержать дополнительные разряды, которые уточняют правила работы с каждым банком: доступен ли банк по записи или/и по чтению, включить или выключить его схемы контроля и т. д.

Если банки выполнены в виде ПЗУ, то адрес регистра (доступного только по записи) может совпадать с адресом одной из ячеек области адресного пространства, отведенной для размещения в ней банков [161]. Например, если банки ПЗУ «вставляются» в область 8000—BFFF адресного пространства микроЭВМ, то регистру можно присвоить адрес 8000. Этот адрес будет соответствовать двум разным элементам — регистру и ячейке ПЗУ. При выполнении операции считывания по адресу 8000 информация будет выдана в шину данных только из ячейки ПЗУ, так как схема выдачи информации в эту шину из регистра не предусмотрена. При выполнении операции записи по этому адресу информация загружается в регистр, а ПЗУ остается в пассивном состоянии, так как оно не срабатывает при наличии сигиала Запись.

Этот прием может быть полезным и в иных случаях, когда не-

обходимо экономить адресное пространство. Например, адреса портов ввода (доступных только по чтению) и вывода (доступных только по записи) можно выбрать одинаковыми. При этом сигнал на управляющей линии Запись служит как бы дополнительным разрядом адреса, выбирающим тот или иной порт.

Метод виртуальной памяти. Применяя методы непосредственного расширения памяти микроЭВМ, следует помнить, что быстродействующая память не может наращиваться «безгранично» в основном из-за увеличения ее стоимости, потребляемой мощности, а также из-за снижения ее быстродействия.

Использование внешних накопителей на магнитных лентах, дисках, запоминающих устройств на цилиндрических магнитных доменах и других устройств подобного класса позволяет увеличивать память машины до нескольких десятков или сотен миллиардов бит при вполне приемлемой стоимости и потребляемой мощности. Конечно, время обмена информацией с такими накопителями на несколько порядков выше, чем время обмена с ячейкой памяти внутреннего ОЗУ или ПЗУ.

Метод виртуальной памяти обеспечивает иллюзию «безграничной» оперативной памяти при вполне ограниченной ее физической емкости (например, 256К ячеек). Идея метода проста: в случае отсутствия в ОЗУ нужной информации в него «подкачивается» необходимая страница из внешнего накопителя, вытесняя одну из «старых» (давно не использовавшихся) страниц. Размер страницы может быть равен, например, 16К ячеек. Механизм «подкачки» страниц поддерживается специальными аппаратными и программными средствами, скрытыми от пользователя. Поэтому пользователь считает, что в его распоряжении имеется оперативная память весьма значительной емкости, которая определяется разрядностью адресной шины.

Предположим, что процессор оперирует с адресным пространством 2^{32} ячеек, используя 32-разрядную адресную шину. Адреса, поступающие из процессора, называются логическими адресами. С точки зрения пользователя к процессору подключена обычная быстродействующая оперативная память с временем доступа менее одной микросекунды и емкостью $2^{32} \approx 4{,}29$ млрд ячеек. Вполне понятно, что создание памяти такой емкости и быстродействия при приемлемой стоимости и потребляемой мощности — задача, технически не осуществимая, по крайней мере, при существующей технологии. Да в этом и нет особой необходимости, в чем мы и убедимся.

Логический 32-разрядный адрес можно представить состоящим из двух частей: старшие 18 разрядов определяют номер N одной из $2^{18}=256$К страниц, с которой процессор собирается работать, а 14 младших разрядов M определяют одну из $2^{14}=16$К ячеек на выбранной странице. Другими словами, в распоряжении пользователя имеются 256К страниц памяти, каждая из которых содержит 16К ячеек. На рис. 1.40 показана укрупненная функциональная схема аппаратного ядра виртуальной памяти. (Виртуальная память реализуется в виде совокупности аппаратных и программных средств: устройств ввода-вывода с соответствующими контроллерами, программ замены страниц и т. д.) На рис. 1.40 показаны следующие устройства, соединенные с магистралью связями, помеченными знаками *, для отражения факта их программной доступности: регистр

логического адреса, ассоциативное запоминающее устройство АЗУ, буферная память ОЗУ1, табличная память ОЗУ2.

Регистр логического адреса предназначен для хранения адреса, поступающего из процессора по магистрали. Ассоциативное запоминающее устройство содержит 16 ячеек, каждая из которых состоит из трех групп разрядов, или *полей*: поля номера логической страницы S (18 бит), поля номера локальной страницы L (4 бита) и поля признаков P, характеризующих некоторые особенности соответствующего массива информации (страницы размеров 16К ячеек), такие, как возможность доступа по записи, факт недавнего использования, факт изменения содержимого и др.

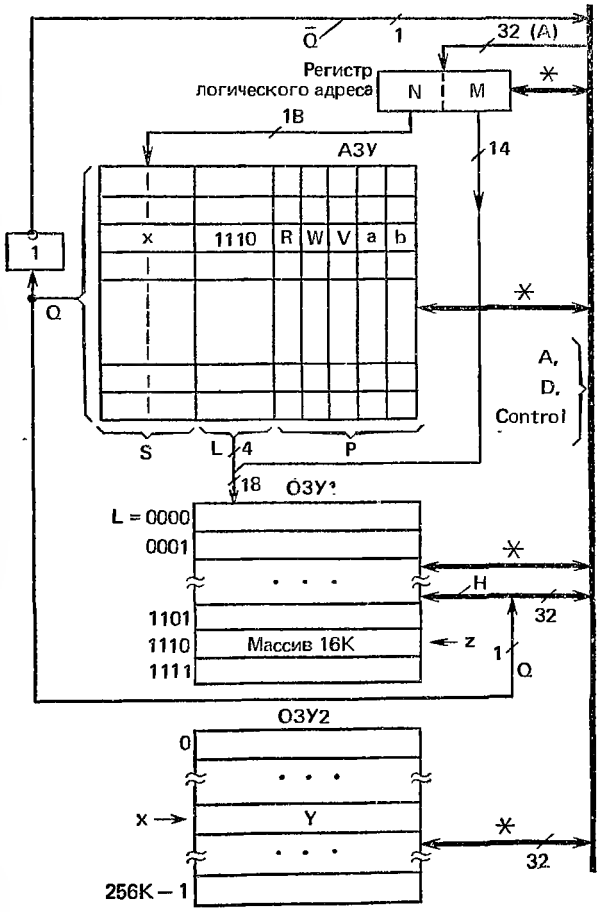


Рис. 1.40. Схемная реализация «ядра» виртуальной памяти

Условием выдачи информации из некоторой ячейки АЗУ является совпадение информации, поступившей из поля N регистра логического адреса, с информацией, хранившейся в поле S этой ячейки. При выявлении такого совпадения в любой из 16 ячеек на выходе Q АЗУ формируется лог. 1, в противном случае — лог. 0.

Память ОЗУ1 организована в виде 16 «локальных» страниц с номерами 0000_2, 0001_2, ..., 1111_2. Каждая страница содержит 16К 32-разрядных ячеек. На вход ОЗУ1 поступает 18-разрядный адрес: четыре старших разряда поступают из поля L выбранной (при совпадении) ячейки АЗУ и определяют номер одной из 16 локальных страниц; 14 младших разрядов — из поля M регистра логического адреса, они определяют адрес ячейки в пределах выбранной страницы. Память ОЗУ1 подключается к магистрали через связи H только в том случае, когда Q=1, т. е. при условии, что в АЗУ выбрана (по сравнению) одна из 16 ячеек. Память ОЗУ2 организована в виде 256К 32-разрядных ячеек и содержит таблицу соответствия логических страниц физическим.

Все перечисленные устройства являются программно-доступными. Это понятие было детально рассмотрено в § 1.2—1.4, поэтому на рис. 1.40 не показана соответствующая аппаратура дешифрации. Вместо этого введены знаки *. Будем считать, что связи, помеченные этими знаками, обеспечивают программный доступ к регистру логического адреса и к любой ячейке памяти АЗУ, ОЗУ1 и ОЗУ2. Наличие программно-доступных элементов, строго говоря, уменьшает число страниц, доступных пользователю. Однако это уменьшение пренебрежимо мало и в данном случае составляет 0,01 % (из 4,29 млрд ячеек исключаются около 525 тыс.).

Инвертор формирует на своем выходе сигнал $\overline{Q}=1$ в том случае, когда ни в одной из 16 ячеек АЗУ не произошло совпадения 18-разрядного входного слова N с 18-разрядным полем S. Сигнал $\overline{Q}$ передается по управляющей шине магистрали в процессор микроЭВМ, оповещая его об отсутствии в ОЗУ1 нужной страницы. На рис. 1.40 не показаны многие детали: схемы стробирования, схемы регенерации памяти, магистральные приемопередатчики и т. п. Их описание потребовало бы дополнительных пояснений, которые не имеют прямого отношения к сути данного метода расширения адресного пространства. Поэтому работа микроЭВМ с виртуальной памятью будет рассмотрена без излишней детализации.

Предположим, что процессор микроЭВМ обратился к памяти по некоторому 32-разрядному адресу, состоящему, как отмечалось, из двух частей: N и M. Пусть далее N=x, а M=z, где x и z — некоторые конкретные коды; x — 18-разрядный код и z — 14-разрядный код. Полученный 32-разрядный адрес запоминается в регистре логического адреса, после чего производится пуск ассоциативной памяти для выяснения вопроса: находится ли страница, логнческий номер которой равен x, в буферной памяти ОЗУ1?

Предположим сначала, что этот вопрос решен положительно, т. е. что предыстория системы такова, что в буферной памяти ОЗУ1 уже размещена нужная страница. Она, как показано на рис. 1.40, расположена в предпоследней зоне (L=1110) размером 16К блока памяти ОЗУ1. Если это так, то одна из записей в АЗУ должна отражать этот факт. Действительно, в третьей сверху ячейке АЗУ содержится запись, указывающая, что логической странице с номером x соответствует локальная страница памяти ОЗУ1, номер кото-

рой равен 1110_2. Поэтому пуск АЗУ приведет, во-первых, к выдаче сигнала Q=1, который подключит ОЗУ1 к магистрали по связям H, и, во-вторых, к выдаче кода 1110 в старшие адресные линии ОЗУ1. После этого производится пуск ОЗУ1, что приводит к выбору нужной ячейки памяти z на странице, локальный номер которой равен 1110, что и требовалось. Содержимое этой ячейки либо обновляется (при записи), либо выдается в шину данных (при чтении). Таким образом, аппаратура позволила быстро выяснить, что нужная страница находится в ОЗУ1, и произвести обращение к нужной ячейке памяти на этой странице.

При последующих обращениях процессора микроЭВМ к странице x (если ее еще не «положили» на свое старое место, например на магнитную ленту) процесс повторяется: АЗУ указывает, где находится эта страница в ОЗУ1, а ОЗУ1 отыскивает нужную ячейку и производит с ней нужную операцию — чтение или запись. Другими словами, до тех пор пока нужная страница находится в ОЗУ1, доступ к ячейке производится с той же скоростью, что и доступ к ячейке «обычного» ОЗУ. Время срабатывания АЗУ «скрадывается»: пока идет выборка из АЗУ, ОЗУ1 может, не дожидаясь результатов работы АЗУ, «на всякий случай» проводить дешифрацию младших 14 разрядов адреса, которые внутри ОЗУ1 могут рассматриваться как старшие разряды. (Такая возможность «превращения» младших разрядов в старшие, а в общем случае — любого разряда в любой имеется, см. § 7.1.)

Предположим теперь, что в ОЗУ1 нет нужной страницы. Это означает, что ни одна из 16 ячеек АЗУ не содержит в поле S код x. Поэтому при обращении к АЗУ будет сформирован сигнал Q=0 и ОЗУ1 вследствие разрыва связей H этим сигналом окажется изолированным от магистрали, хотя в процессор микроЭВМ и будет послан ответный сигнал (чтобы не было «зависания»). Пройдя через инвертор, сигнал отсутствия совпадения поступает в процессор микроЭВМ, вызывая прерывание выполнения программы в связи с отсутствием нужной страницы в ОЗУ1. Процессор временно откладывает текущее задание и переходит к выполнению прерывающей программы. Задача этой программы — отыскать, где находится физическая страница, соответствующая коду x, и ввести эту страницу в ОЗУ1, сделав соответствующую пометку в АЗУ. Эта страница вводится на свободное место в ОЗУ1, а если такового нет — на место той страницы, которая дольше всего не использовалась.

Отыскание номера физической страницы, соответствующей данной логической странице, производится следующим образом. Процессор считывает содержимое программно-доступного регистра логического адреса, выделяет из полученного 32-разрядного кода поле N (в котором записан код x) и использует его в качестве младших разрядов адреса при обращении к ОЗУ2, в котором хранится таблица соответствия логических номеров страниц местам их хранения на внешних носителях. Обратившись к ячейке x ОЗУ2, процессор получит в ответ некоторый 32-разрядный код Y, который является указателем физической страницы. В этом коде содержится информация о том, где находится эта страница. В нашем примере она находится в определенной зоне магнитной ленты одного из накопителей.

Теперь процессор должен определить, на какое место в ОЗУ1 можно ввести нужную страницу из внешнего накопителя. Для этого он считывает поле признаков P АЗУ по всем 16 ячейкам и, анали-

зируя специально выделенные биты этого поля, определяет (с учетом предыстории изменения этих битов), какие ячейки АЗУ дольше всего оказывались невостребованными. Определив, что давно не было обращения к странице 1110_2 ОЗУ1, процессор решает ликвидировать старую запись на этой странице. Поэтому он вместо старой строки записывает в АЗУ новую (см. третью сверху строку АЗУ, рис. 1.40), которая говорит о том, что логической странице x будет теперь соответствовать локальная страница 1110 ОЗУ1.

После этого, используя обычный программный доступ к ОЗУ1, процессор вводит нужную страницу с магнитной ленты в предпоследнюю зону ОЗУ1 размером 16К ячеек. На этом прерывающая программа заканчивает работу по «подкачке» нужной страницы в ОЗУ1 и управление передается команде, которую ранее не удавалось выполнить из-за отсутствия страницы в ОЗУ1. Случаи замены страниц тем более редки, чем больше размер страницы и чем больше их хранится в буферном ОЗУ (ОЗУ1).

Определение давности использования страниц в простейшем случае может производиться путем периодического программного опроса и сброса признаков R во всех ячейках АЗУ [9], например, по сигналам от сетевого таймера частотой 50 Гц. При обращении к ячейке АЗУ (при совпадении кодов N и x) признак R в данной ячейке автоматически устанавливается в единицу. Поэтому если периодически опрашивать признаки R и сбрасывать их, то можно вести учет актуальности страниц. Страницы, утратившие свою актуальность, не будут восстанавливать признаки R после сброса. Страницы, к которым производятся обращения, будут восстанавливать признаки R к моменту очередного опроса. (Эта процедура напоминает периодическую «перекличку» группы людей — те, кто долго не «отмечался», могут быть исключены из сообщества и заменены более дисциплинированными «со стороны».) Просматривая таблицу учета признаков R, процессор видит, что старые страницы «давно» не восстанавливали признаки R, средние — восстанавливали в недалеком прошлом, но перестали восстанавливать в последнее время, новые страницы пока еще актуальны, так как в самой недавней предыстории они восстанавливали признаки R. Таким образом, процессор имеет возможность выбрать наиболее устаревшую страницу для ее уничтожения.

Простое уничтожение страницы в ОЗУ1 возможно лишь в том случае, когда ее содержимое не было изменено в процессе работы с ней. В этом случае точная копия страницы хранится во внешнем накопителе (в нашем примере — на магнитной ленте). Однако не исключена возможность того, что в процессе работы со страницей, вызванной ранее из внешнего накопителя в ОЗУ1, было изменено содержимое хотя бы одного бита в этой странице. В этом случае, прежде чем уничтожить страницу, необходимо переслать ее обратно во внешний накопитель для приведения в точное соответствие хранящегося на магнитной ленте оригинала и откорректированной копии.

Если пользователю необходимо сохранить старую (неизменную) версию информации на этой странице, то он должен позаботиться об этом сам: пользовательская программа должна заблаговременно снять копию старой страницы и поместить ее в «надежном» месте. Чтобы узнать, производилась ли операция записи на страницу (это принимается за критерий ее изменения), в поле P вводится бит W,

который автоматически устанавливается в единичное состояние при выполнении операции записи иа эту страницу. При решении вопроса о необходмиости возврата уничтожаемой страницы на внешний носитель, например на магнитную ленту, процессор руководствуется значением бита W: при W=0 копия в точности соответствует оригиналу и переписи не требуется.

В поле P имеется также признак V, единичное состояние которого говорит о том, что страница активна. Уничтожение страницы производится переводом признака V в нулевое состояние.

Еще одна особенность виртуальной памяти — встроенный механизм защиты страниц, содержащихся в буферной памяти, от несанкциоиированного доступа. Согласно [9], например, используются два бита a и b, которые включаются в поле P и определяют четыре уровня защиты страницы, как показано в табл. 1.9.

Таблица 1.9

a	b	Режим операцнонной системы	Режнм пользователя
0	0	Только чтение	Недоступна
0	1	Чтение и запись	»
1	0	»	Только чтение
1	1	»	Чтение и запись

При a=0 пользовательские программы не имеют права доступа к странице. При попытке такого доступа (из-за ошибки программиста или по его «злому умыслу») в процессор посылается сигнал прерывания и он, отложив выполнение пользовательской программы, предпринимает соответствующие действия, например, выдает сообщение оператору, перезагружает программу и пускает ее вновь и т. д. Самый «легкий» режим — полное отсутствие защиты — соответствует случаю a=1, b=1. Схемы защиты памяти на рис. 1.40 не показаны.

Виртуальная память первоначально была реализована на больших ЭВМ и мини-ЭВМ. Однако по мере расширения сферы применения микроЭВМ и увеличения степени интеграции элементной базы возникла необходимость в реализации виртуальной памяти и для этого класса машин. Например, в работе [9] рассмотрены вопросы реализации виртуальной памяти для микроЭВМ с использованием БИС серии NS 16000.

Глава 2

Обеспечение связи микропроцессорной системы с «внешним миром»

2.1. Интерфейс «общая шина» (упрощенная версия)

Термин *интерфейс* обычно используется для обозначения совокупности правил взаимодействия некоторой группы компонентов системы и правил построения средств, обеспечивающих реализацию этого взаимодействия. Далее рассмотрен интерфейс «общая шина» как один из наиболее простых и «логичных». Степень детализации описания выбрана достаточной для последующего рассмотрения принципиальных схем контроллеров различных типов.

В каждом акте обмена участвуют два устройства: активное и пассивное. Определены три типа обменов: запись, чтение и прерывание. Перед тем как произвести обмен, активное устройство должно захватить магистраль с тем, чтобы никто из конкурентов не помешал ему работать с ней.

Захват магистрали. Для описания механизма захвата магистрали рассмотрим архитектурный элемент микроЭВМ — арбитр, который ранее (см., например, рис. 1.4) был скрыт в одном из блоков, подключенных к магистрали. На рис. 2.1 кружками обозначены устрой-

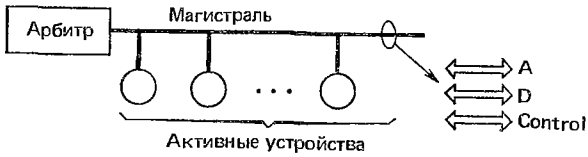


Рис. 2.1. Подключение арбитра к магистрали

ства, способные стать активными. Именно они представляют сейчас для нас интерес, поскольку способны бороться за право пользования магистралью. Арбитр является судьей в этой борьбе. Та же схема в более подробном виде представлена на рис. 2.2. Активные устройства расположены в виде матрицы, что, вообще говоря, является отступлением от «магистрального» принципа. Но можно «успокоить себя» тем, что если все активные устройства расположить в один ряд, а провода «связать в жгут», то мы вернемся к схеме, показанной на рис. 2.1.

Обозначения: BR_i — управляющие линии для передачи сигналов запросов в арбитр ($i=0, 1,...,7$); BG_i — управляющие линии для передачи сигналов разрешений из арбитра к активным устройствам; SACK — линия передачи сигнала ответа от первоочередного претендента на подключение к магистрали; BBSY — линия для передачи сигнала занятости магистрали; R — вход сброса арбитра. Эти же сокращения будут использованы для обозначения сигналов на линиях.

Активные устройства в данной версии интерфейса разбиты на восемь групп. Максимальный приоритет имеют устройства верхней (нулевой) группы. С увеличением номера группы ее приоритет уменьшается. В пределах группы приоритет убывает по мере удаления от арбитра. Максимальным приоритетом обладает верхнее левое активное устройство, а минимальным — нижнее правое.

Активные устройства по собственной инициативе и независимо друг от друга формируют сигналы запросов (напряжения низкого уровня) на подключение к магистрали. Эти сигналы складываются по схеме Монтажное ИЛИ на запросных линиях BR_i и поступают

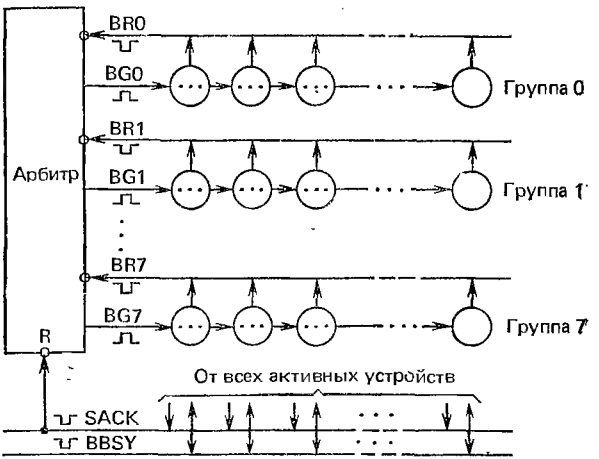


Рис. 2.2. Подключение активных устройств к арбитру

в арбитр. (На запросной линии устанавливается напряжение низкого уровня при наличии хотя бы одного запросного сигнала.) Арбитр воспринимает запросные сигналы и выдает сигнал разрешения по одной из линий BG_i. При наличии конкурирующих групп разрешение выдается группе с меньшим номером. При наличии конкурирующих устройств в пределах одной группы разрешение получает устройство, расположенное ближе к арбитру. Это устройство запрещает дальнейшую передачу сигнала разрешения.

Линия SACK предназначена для оповещения арбитра о том, что сформированный им сигнал разрешения дошел до цели, т. е. до одного из запросивших магистраль устройств. При получении сигнала SACK арбитр прекращает выдачу сигнала разрешения. Линия BBSY предназначена для оповещения всех активных устройств (точнее, очередника) о том, что магистраль еще не освободилась. После ее освобождения очередник занимает ее и сам формирует сигнал BBSY. При этом очередник снимает сигнал SACK и, пока в магистрали идет обмен, арбитр готовит нового очередника и т. д. Таким образом, фаза обмена совмещена по времени с фазой поиска очередника. На

рис. 2.3 представлена временна́я диаграмма процесса захвата магистрали.

В момент t_0 «наше» активное устройство формирует на запросной линии своей группы напряжение низкого уровня — сигнал запроса, который поступает в арбитр. В момент t_1 на вход «нашего» активного устройства поступает сигнал разрешения в виде напряжения высокого уровня. Сигналы разрешений передаются напряжением высокого уровня в отличие от всех остальных сигналов «общей шины». «Наше» активное устройство поглощает сигнал разрешения, т. е. препятствует его дальнейшему распространению в группе, и оповещает арбитр о его получении выдачей напряжения низкого уровня в линию SACK в момент t_2. Арбитр «узнаёт», что сигнал разрешения

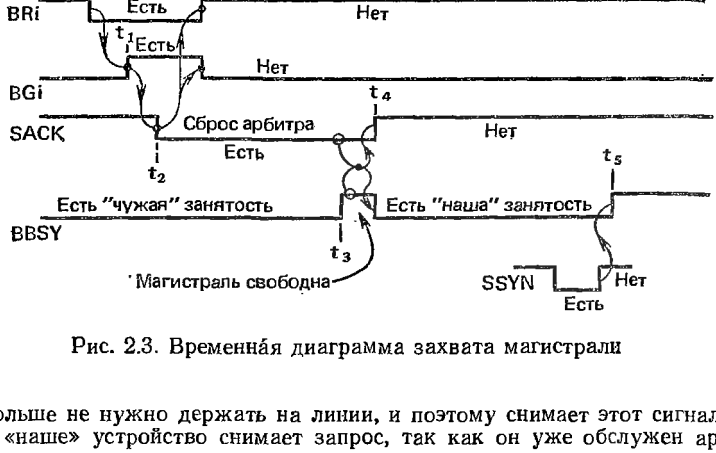


Рис. 2.3. Временна́я диаграмма захвата магистрали

больше не нужно держать на линии, и поэтому снимает этот сигнал, а «наше» устройство снимает запрос, так как он уже обслужен арбитром.

«Наше» активное устройство готово захватить магистраль, однако в данном примере она все еще занята «чужим» активным устройством, которое ведет обмен. В момент t_3 магистраль освобождается. «Наше» активное устройство, зная, что именно оно держит на линии SACK напряжение низкого уровня, и «увидя», что магистраль освободилась от «чужого» обмена, само формирует напряжение низкого уровня на линии BBSY и приступает к обмену с выбранным абонентом, одновременно снимая сигнал с линии SACK (момент t_4). Начиная с этого момента, арбитр воспринимает новую «порцию» запросных сигналов и готовит следующего очередника, а «наше» устройство ведет обмен. По окончании ответного сигнала SSYN от абонента наше» устройство освобождает магистраль для очередника (момент t_5).

Временна́я диаграмма записи. Операция записи заключается в том, что активное устройство, захватив магистраль, пытается передать данные в один из программно-доступных элементов системы, например в ячейку памяти ОЗУ. При отсутствии ошибок в указании адреса такая попытка окажется успешной, в противном случае при обращении по несуществующему адресу произойдет «зависание». На

рис. 2.4 приведена временна́я диаграмма записи при правильном указании адреса. Последовательность событий следующая.

Момент t_1. Захватив магистраль, активное устройство выдает сигнал по линии C1 (запись), адрес и данные, которые нужно записать по этому адресу.

Момент t_2. Через время, не меньшее 150 нс (75 нс — на «успокоение» магистрали и 75 нс — запас), активное устройство выдает сигнал сопровождения адреса MSYN.

Момент t_3. Одно из устройств, например ОЗУ, опознаёт адрес и принимает информацию с шины данных в соответствующую ячейку. Приняв данные, это устройство отвечает ведущему сигналом SSYN.

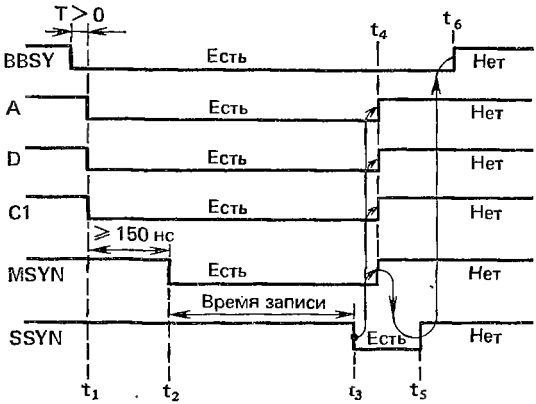


Рис. 2.4. Временна́я диаграмма записи

Момент t_4. Получив сигнал SSYN, активное устройство «узнаёт», что запись состоялась, и поэтому снимает с магистрали все сигналы, кроме сигнала BBSY.

Момент t_5. Убедившись в том, что ответный сигнал SSYN воспринят активным устройством (о чем говорит снятие сигнала MSYN), ОЗУ снимает сигнал SSYN.

Момент t_6. Активное устройство, дождавшись снятия сигнала SSYN, освобождает магистраль, т. е. снимает сигнал BBSY.

При зависании ни один из абонентов не отвечает на обращение со стороны активного устройства, так как обращение производится в свободную область адресного пространства. В этом случае активное устройство выждав примерно 10 мкс, искусственно завершает обмен, т. е. выдает «само себе» сигнал SSYN, снимает сигналы A, D, C1, MSYN, SSYN и затем BBSY. В зависимости от «интеллекта» активного устройства факт зависания либо игнорируется, либо принимается во внимание. Если, например, активным устройством является процессор, то следствием зависания может быть переход к программе, которая анализирует причины зависания и выдает оператору текстовое сообщение о случившемся. Однако если эта программа исказились и процессор, выйдя на нее, вновь зависает, то он перехо-

дит в режим Останов, так как без помощи человека-оператора он не может выйти из создавшейся ситуации.

Временна́я диаграмма чтения. При чтении активное устройство пытается получить слово данных от некоторого программно-доступного элемента, например от ячейки ОЗУ. При правильном указании адреса реализуется временна́я диаграмма, показанная на рис. 2.5. При отсутствии абонента с указанным адресом возникает зависание, которое примерно через 10 мкс устраняется искусственной выдачей ответного сигнала от специальной схемы, как и в предыдущем случае. Приведем последовательность событий при чтении.

Момент t_1. Захватив магистраль, активное устройство выдает адрес ячейки памяти, из которой нужно извлечь данные (одно слово).

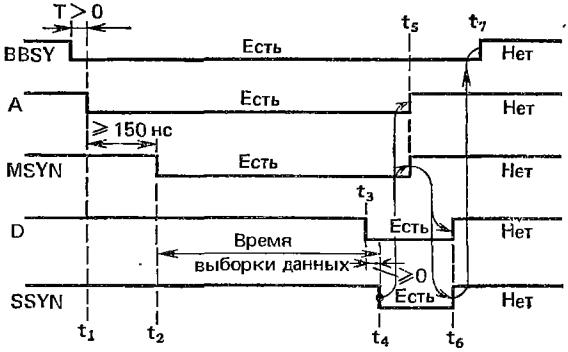


Рис. 2.5. Временна́я диаграмма чтения

Момент t_2. Активное устройство формирует сигнал MSYN, подтверждающий истинность адреса. Напряжение на линии C1 постоянно равно примерно +3,5 В, что расценивается всеми устройствами как режим чтения.

Момент t_3. Схемы дешифрации адреса выбирают нужную ячейку ОЗУ. Информация из этой ячейки по шине данных передается активному устройству.

Момент t_4. Активное устройство, получив сигнал SSYN, узнаёт о том, что через время, большее 75 нс, информацию можно будет принять с шины данных. Выждав 75 нс (или более), устройство принимает информацию. Таким образом, компенсацию времени «успокоения» шины данных ведут активные, а не пассивные устройства. Это разумно, поскольку обычно пассивных устройств в системе больше, чем активных, и в каждом из них пришлось бы предусматривать элемент задержки.

Момент t_5. Приняв информацию, активное устройство снимает адрес и сигнал MSYN, так как чтение уже состоялось и эти сигналы утратили актуальность.

Момент t_6. Оперативное запоминающее устройство снимает данные и сигнал SSYN, так как оно узнало (по снятию сигнала с линии MSYN) о том, что данные уже приняты активным устройством.

Момент t_7. Активное устройство освобождает магистраль для «очередника».

Режим Чтение — модификация — запись по существу является комбинацией двух предыдущих: активное устройство сначала производит чтение информации по некоторому адресу, а затем, после некоторой паузы, связанной с обработкой прочитанной информации,— запись новой информации по тому же адресу. На протяжении всего этого двойного цикла адрес передается активным устройством только один раз — в начале цикла, а магистраль занята им постоянно.

Этот режим позволяет ускорять выполнение команд, связанных с преобразованием информации, размещенной в ячейках памяти. Временные диаграммы, соответствующие этому режиму, схожи с рассмотренными ранее, поэтому здесь не приводятся. Однако тот факт, что при чтении, преобразовании и записи информации магистраль постоянно занята активным устройством, имеет большое «архитектурное» значение и заслуживает более детального рассмотрения. Речь пойдет о так называемых программных семафорах — своего рода программных арбитрах, регулирующих очередность обращения нескольких независимых активных устройств к общему «ресурсу».

Предположим, что одна из ячеек памяти, например ячейка с адресом 579Е, используется как указатель занятости некоторого ресурса, так же как «настоящий» семафор разрешает или запрещает пользование участком железнодорожного пути. Таким ресурсом может быть определенная область памяти, контроллер, вспомогательный процессор и т. д. Предположим далее, что нулевой код в ячейке 579Е соответствует свободному ресурсу, а ненулевой — занятому. В исходном состоянии ресурс свободен.

При работе системы не исключена возможность того, что некоторые активные устройства А и В одновременно или почти одновременно пожелают воспользоваться общим ресурсом. Благодаря наличию схемного магистрального арбитра (см. рис. 2.2) строго одновременное обращение к семафору с целью выяснения его состояния запрещено. Однако этого может оказаться недостаточно для правильной работы системы, если в ней используются только обычные процедуры чтения и записи. (Строго говоря, можно обойтись и без использования режима Чтение — модификация — запись, см. § 7.25, однако при этом программный арбитраж осуществляется слишком медленно.) Действительно, в отсутствие режима Чтение — модификация — запись могла произойти следующая последовательность событий:

1. Устройство А занимает магистраль, обращается в режиме чтения к ячейке 579Е и, получив из нее код, освобождает магистраль.

2. Устройство А сравнивает полученный из ячейки 579Е код с нулевым кодом. В это время устройство В занимает магистраль, обращается в режиме чтения к ячейке 579Е и, получив из нее код (тот же, что и прочитанный ранее устройством А), освобождает магистраль.

3. Устройство А узнаёт о том, что ресурс свободен, поскольку ранее прочитанное им содержимое ячейки памяти с адресом 579Е оказалось равным нулю. В это время устройство В сравнивает полученный ранее код с нулевым кодом.

4. Устройство А, считая, что ресурс находится в его распоряжении, захватывает магистраль, засылает ненулевой код в ячейку 579Е и затем освобождает магистраль, готовясь воспользоваться ресурсом. В это время устройство В узнаёт о том, что ресурс свободен

(сведения, на которых основан этот вывод, как видим, устарели).

5. Устройство А приступает к пользованию ресурсом. Устройство В, ошибочно считая, что ресурс находится в его распоряжении, засылает ненулевой код в ячейку 579Е и в дальнейшем также приступает к пользованию уже занятым ресурсом, что недопустимо.

Из этого примера видно, что если бы устройство А занимало магистраль вплоть до окончания анализа состояния семафора и его перевода в запрещающее состояние, устройство В не получило бы ложных сведений о состоянии семафора. Иными словами, при использовании режима Чтение — модификация — запись конфликты не возникают, поскольку первое же активное устройство, которое увидело разрешающее состояние семафора, переводит его в запрещающее состояние в том же временно́м цикле, не отключаясь от магистрали. Конкурирующие активные устройства в этом случае уже не смогут, как в только что рассмотренном примере, увидеть разрешающее состояние семафора, так как этим состоянием только что поинтересовался более удачливый конкурент.

Для работы с семафорами во многих микроЭВМ предусмотрена команда типа «проверить и установить», которая может выполняться следующим образом. Процессор считывает содержимое ячейки — семафора и, если считан нулевой код, устанавливает в своем регистре состояния признак Нуль. Выполнение команды завершается записью кода 111...1 в ячейку — семафор независимо от результата считывания (если были считаны нули, то семафор переводится в запрещающее состояние, если были считаны единицы, то запрещающее состояние подтверждается). На протяжении считывания и записи магистраль постоянно занята процессором. После команды «проверить и установить» в программе записана команда ветвления по признаку Нуль регистра состояния процессора, так что процессор принимает решение о возможности использования ресурса.

При отсутствии такой команды в системе команд микроЭВМ задача управления семафором представляется более сложной и на первый взгляд может даже показаться неразрешимой. Например, используя систему команд мнкроЭВМ «Электроника-60», многие далеко не сразу отыскивают правильные варианты решения. Приведем один из них (описание программы совмещено с описанием работы системы).

1. Активное устройство выполняет операцию чтения содержимого ячейки памяти S, используемой как семафор.

2. Активное устройство сравнивает полученный код с кодом 077777_8, принятым в качестве указателя доступности ресурса (все отличные от указанного коды характеризуют занятое состояние ресурса).

3. Если ресурс занят, то активное устройство вновь переходит к действиям по п. 1. Если ресурс «вроде бы» свободен (точной гарантии его освобождения пока еще нет), то активное устройство переходит к действиям по п. 4.

4. Активное устройство выполняет команду прибавления единицы к коду, хранящемуся в ячейке S. Эта команда реализуется с использованием режима Чтение — модификация — запись. Если в качестве операнда для сложения был использован код 077777, то признак V арифметического переполнения в регистре состояния процессора (активного устройства) устанавливается в 1. Во всех остальных случаях признак V устанавливается в 0.

5. Активное устройство выполняет команду условного перехода

по результатам анализа признака V. Если V=0 (ресурс при окончательной проверке оказался уже занятым), то производится переход к п. 1. Если V=1 (ресурс при окончательной проверке оказался свободным), то производится переход к п. 6.

6. Активное устройство использует ресурс (или предоставляет его кому-либо другому, по своему усмотрению, проявляя некую машинную «гуманность победителя»).

7. Активное устройство (или его «протеже») освобождает ресурс и засылает код 077777 в ячейку S, открывая семафор. После этого устройства-претенденты на использование ресурса вступают в очередной тур конкурентной борьбы.

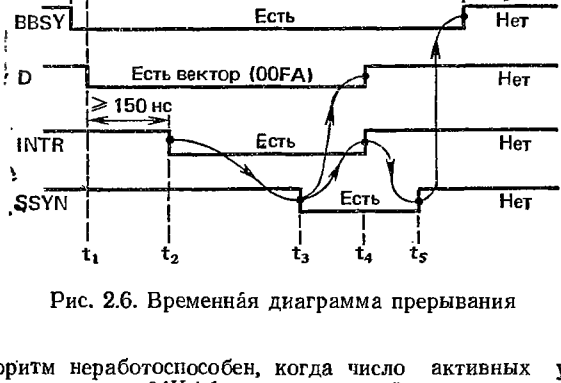


Рис. 2.6. Временна́я диаграмма прерывания

Алгоритм неработоспособен, когда число активных устройств равно или превышает 64K+1 и все эти устройства выстроились в очередь к ресурсу. В этой (нереальной) ситуации в ячейке S будет наблюдаться «переполнение», вызванное прибавлением большого числа единиц к содержимому этой ячейки.

Недостатком этого и подобных алгоритмов является то, что активные устройства, которые находятся в состоянии ожидания освобождения ресурса, выполняют действия по пп. 1—3 и занимают магистраль «бесполезными» пересылками, создавая ее «закупорку». (Как с этим бороться, показано в § 7.31.)

Временна́я диаграмма прерывания. Механизм прерывания предназначен для оповещения процессора микроЭВМ о том, что во внешнем мире произошло некоторое событие, существенное для ее дальнейшего поведения.

Это событие может состоять в том, что пользовательская программа «просит» операционную систему ввести или вывести массив информации. Это — так называемое программное прерывание. Его выполнение не сопровождается появлением каких-либо специфических сигналов на линиях магистрали. Поэтому нас будет интересовать другой тип прерываний, который обусловлен срабатыванием схем связи с сигнальными линиями внешнего мира и называется аппаратным прерыванием. На рис. 2.6 приведена временна́я диаграмма аппаратного прерывания.

Для определенности будем считать, что в обмене участвуют два

устройства: активным устройством является контроллер электрической пишущей машинки, а пассивным — процессор.

В некоторый момент контроллер просит у процессора обслуживания, например, в связи с необходимостью возврата каретки. (Можно было бы предусмотреть пружинно-рычажный механизм для автоматического возврата каретки. Однако красота микропроцессорной техники в том и заключается, чтобы упростить объект, убрать все лишние механические детали. Ведь процессору при его высоком интеллекте ничего не стоит временно отвлечься от выполняемого задания и выполнить столь незначительную просьбу.) Цель контроллера в данном случае состоит в том, чтобы временно прервать работу процессора и отослать его к соответствующей программе, адрес начала которой известен и хранится, например, в ячейке 00FA ОЗУ. В следующей ячейке — 00FB — хранится новое содержимое регистра состояния процессора.

Код 00FA является так называемым вектором прерывания — косвенным адресом начала программы. (В другой, более громоздкой терминологии этот код называется адресом вектора прерывания, но суть от этого не меняется.) При выполнении прерывающей программы, на которую процессор вышел по вектору, производится обращение по адресу к программно-доступному электромагниту возврата каретки, после чего процессор возвращается к прерванной задаче, встретив в программе команду *выход из прерывания*. Приведем перечень событий при отработке процедуры прерывания.

Момент t_1. Захватив магистраль, контроллер выставляет в младших восьми разрядах шины данных код FA; старший байт (восемь разрядов) шины данных не используется для передачи вектора. Таким образом, на шине данных устанавливается код 00FA.

Момент t_2. Через время, большее или равное 150 нс, контроллер выдает сигнал сопровождения вектора INTR аналогично выдаче сигнала MSYN при чтении или записи.

Момент t_3. Процессор принимает вектор и отвечает контроллеру сигналом SSYN.

Момент t_4. Контроллер, убедившись в том, что процессор принял вектор, о чем свидетельствует появление сигнала SSYN, снимает вектор и сигнал INTR

Момент t_5. Процессор снимает сигнал SSYN, так как снят сигнал INTR.

Момент t_6. Контроллер, дождавшись снятия сигнала SSYN, освобождает магистраль, т. е. снимает сигнал BBSY.

Таким образом, контроллер выполнил свою задачу: он передал в процессор вектор прерывания и отключился от магистрали; дальнейшая инициатива перешла к процессору.

Мы рассмотрели все три режима работы магистрали — запись, чтение и прерывание. Теперь можно утверждать, что «с точки зрения» магистрали, микроЭВМ, при всем ее высоком интеллекте, умеет лишь читать, писать и реагировать на сигналы прерывания. В этом можно убедиться с помощью осциллографа или логического анализатора, подключив его к магистрали. (При этом поражает однообразие протекающих в магистрали процессов.) Такой упрощенный взгляд на микроЭВМ (равно, как взгляд на автомобиль как на устройство для истирания поршневых колец) позволяет инженеру-схемотехнику отчетливо определять «сферу своего влияния»: его задача сводится к тому, чтобы отработать схемные решения своих контроллеров до

такого уровня, чтобы все три режима выполнялись корректно, без «гонок». Задача, как видим, совсем не сложная.

Начальный пуск процессора и защита вычислений от «провалов» напряжения первичной питающей сети переменного тока. На рис. 2.7 показаны временны́е диаграммы, которыми сопровождается выключение и включение напряжения первичной питающей сети (~220 В, 50 Гц). Приведем описание происходящих при этом событий.

Момент t_1. В результате преднамеренного действия оператора или сбоя напряжение первичной питающей сети переменного тока падает ниже некоторого порогового значения, например ниже 150 В (на рис. 2.7 показано полное исчезновение напряжения).

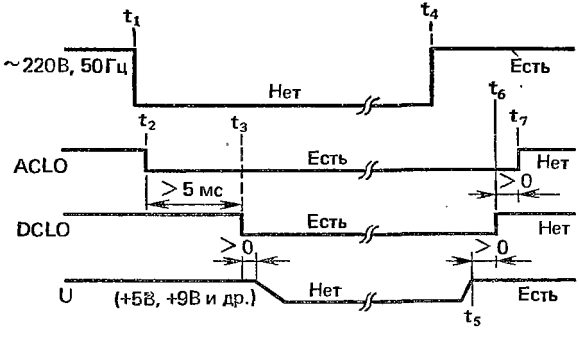


Рис. 2.7. Временна́я диаграмма сигналов при выключении и включении напряжения питания микроЭВМ

Момент t_2. На линии ACLO (авария сетевого питания) формируется напряжение низкого уровня. Эта линия входит в управляющую шину магистрали. Источник сигнала ACLO — блок питания микроЭВМ (если в микроЭВМ несколько блоков питания, то первый выключившийся блок); приемник — процессор и/или какое-либо другое устройство, заинтересованное в «спасении» информации. Начиная с момента t_2 все заинтересованные устройства прекращают выполнять рабочие программы и начинают записывать в определенную область ОЗУ информацию, которая в дальнейшем, при восстановлении напряжения питания, позволила бы им продолжить работу по своим программам, начиная с прерванных мест. Эта область ОЗУ выполнена на статических КМОП БИС, имеющих малое потребление энергии в режиме хранения информации. Время «упрятывания» информации в статическую память в связи с приближающейся аварией вторичного питания (+5; +9 В и др.) не должно превышать 5 мс. Это время достаточно велико, так как цикл обращения к статическому КМОП ОЗУ не превышает 1 мкс. Напряжения источника вторичного электропитания поддерживаются в заданных пределах за счет энергии, накопленной в конденсаторах фильтра.

Момент t_3. На управляющей линии DCLO (авария источника вторичного электропитания) блок питания формирует напряжение низкого уровня, которое рассматривается всеми устройствами как

сигнал общего сброса. Этот сигнал является «последним предупреждением» перед уходом напряжений вторичного питания за допустимые границы. Он закрывает по входам статическое ОЗУ (рис. 2.8), в котором упрятана вся информапия, необходимая для последующего учета предыстории работы микроЭВМ. Шина питания этого ОЗУ через диод подключена к аварийному источнику питания, например к гальванической батарее, поэтому информация в ОЗУ сохраняется и при отсутствии напряжения на выходах вторичного источника питания (БИС ОЗУ хранят информацию при снижении напряжения питания до +3 В). При наличии «штатного» напряжения диод закрыт и энергия от аварийного источника питания не отбирается.

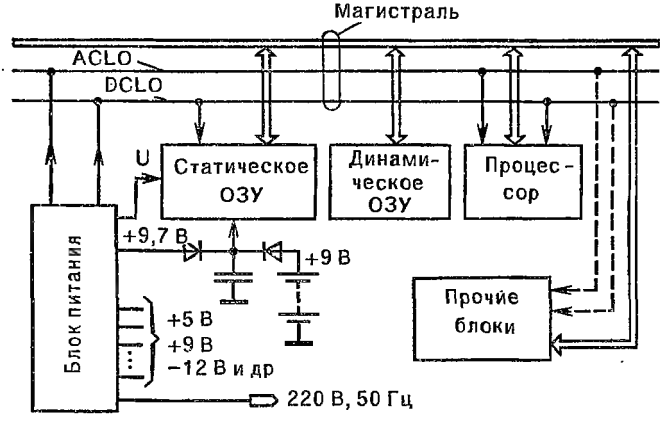


Рис. 2.8. Схема подключения блока питания и внешней батареи к микроЭВМ для обеспечения сохранения информации при прерывании питающего напряжения

После закрытия ОЗУ уже нет опасности искажения его содержимого при исчезновении напряжений вторичного питания. Обесточенное состояние микроЭВМ может поддерживаться достаточно долго — все определяется энергоемкостью аварийного источника питания. Ток потребления статического ОЗУ в этом режиме — единицы миллиампер.

Момент t_4. Напряжение первичной питающей сети достигает номинального значения.

Момент t_5. Напряжения на шинах вторичного питания достигают номинальных значений.

Момент t_6. Сигнал DCLO принимает единичное значение. Статическое ОЗУ открывается для доступа.

Момент t_7. Сигнал ACLO принимает единичное значение. Это событие используется для пуска активных устройств с жестко заданных адресов по своим программам. Каждое из устройств «знает», где была упрятана нужная информация (об этом должен позаботиться программист), поэтому система может продолжать работу без вме-

шательства оператора, если, конечно, решаемая задача допускает такое продолжение.

Отметим некоторые особенности построения микроЭВМ с сохранением информации в статическом ОЗУ с резервным питанием.

1. Программа «спасения» информации, выполняемая в интервале t_2—t_3 (см. рис. 2.7), должна завершиться командой Останов до t_3. Процессор, получив эту команду, уже не проявляет активности на магистрали, и к моменту t_3 все линии магистрали находятся в пассивном состоянии. Это обеспечивает гарантию того, что в момент закрытия ОЗУ по входам не был «сорван» текущий обмен, что могло бы привести к искажению содержимого одной или даже нескольких ячеек памяти.

2. Если микроЭВМ содержит несколько блоков питания, то каждый из них вырабатывает сигналы ACLO и DCLO, которые суммируются с одноименными сигналами на соответствующих линиях магист-

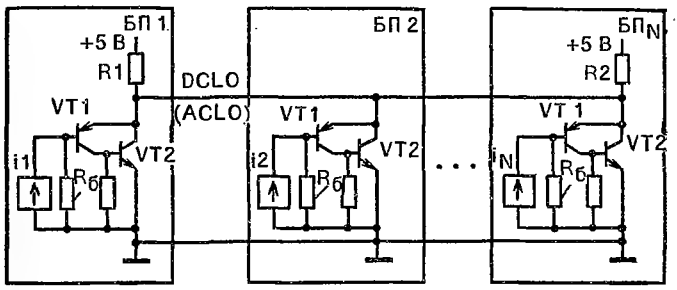


Рис. 2.9. Выходные каскады формирователей сигналов DCLO и ACLO в блоках питания БП1—БП$_N$, обеспечивающие поддержание этих сигналов в состоянии 0 при выключении хотя бы одного блока

рали по схеме Монтажное И. Проблема заключается в том, чтобы сформировать напряжения низких уровней на обеих общих линиях и поддерживать их даже в том случае, когда выключен только один блок питания, а остальные включены. Иными словами, обесточенный блок питания должен оставаться «работоспособным» в том смысле, что он должен гарантировать поддержание напряжений низкого уровня (менее 1 В) на линиях ACLO и DCLO, несмотря на попытку других блоков питания повысить напряжения на этих линиях. (Задача на первый взгляд неразрешимая — ведь выключенный блок питания, казалось бы, неспособен что-либо «гарантировать».)

Решение этой задачи показано на рис. 2.9 [3]. Блоки питания БП1—БП$_N$ соединены с линиями DCLO и ACLO «общей шины». Показана линия DCLO, подключение к линии ACLO производится аналогично. Все линии связи, соединяющие различные составные части микроЭВМ, подключаются на физических концах к согласующим резисторам или резисторным делителям напряжения (см. гл. 4). В данном примере согласующий резистор R1 подключен к шине +5 В бло-

ка питания БП1, а резистор R2 — к шине +5 В блока питания $БП_N$.

Если все блоки питания включены, то работают источники тока $i1—i_N$, падение напряжения на резисторах R_6 во всех блоках питания достаточно велико для поддержания транзисторов VT1 в выключенном состоянии, поэтому транзисторы VT2 также выключены. На линии DCLO поддерживается напряжение высокого уровня (примерно +5 В). Если один из блоков питания, например блок БП2, выклю-

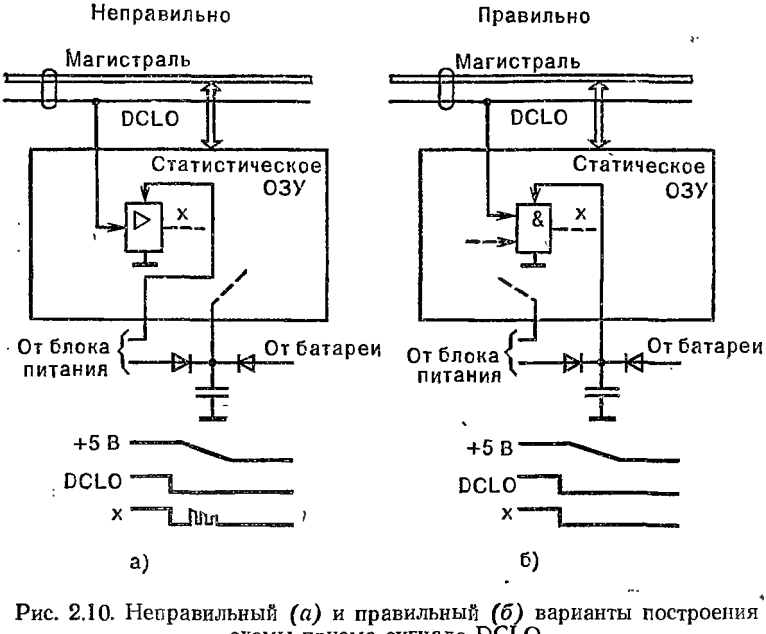


Рис. 2.10. Неправильный (*а*) и правильный (*б*) варианты построения схемы приема сигнала DCLO.

Попытка принять сигнал DCLO через буферный элемент со «штатным» питанием (*а*) приводит к потере информации в ОЗУ; этот сигнал должен приниматься элементом с резервированным питанием (*б*)

чается, то ток i2 прекращается и транзисторы VT1 и VT2 этого блока начинают работать в режиме «стабилитрона» с напряжением стабилизации около 1 В. Поэтому, несмотря на то что блоки питания БП1 и $БП_N$ остались включенными, напряжение на линии DCLO фиксируется на низком уровне. Пример схемы формирования сигналов ACLO и DCLO приведен в [10].

3. При построении схемы запрета работы статического ОЗУ линия DCLO должна подключаться непосредственно к логическому элементу с резервным питанием (рис. 2.10, *б*), так как сигнал на этой линии не подвержен «дребезгу» при исчезновении основного питания. Если этот сигнал пропустить через логический элемент с основным питанием (рис. 2.10, *а*), то в период угасания напряжения на выходе элемента может наблюдаться генерация. Поэтому статическое ОЗУ

не будет защищено от непредсказуемых сочетаний сигналов в магистрали, т. е. не будет гарантии сохранности информации, записанной в ОЗУ процессором.

4. Интервал времени t_6—t_7 (рис. 2.7) обычно используется для приведения БИС динамической памяти в работоспособное состояние. Для правильной работы динамического ОЗУ после включения напряжения питания необходимо провести 16 или более циклов регенерации, каждый из которых длится около 2 мс. Циклы регенерации могут осуществляться схемами управления динамического ОЗУ без участия процессора, так что к моменту t_7 пуска процессора динамическое ОЗУ оказывается готовым к работе.

5. При включении основного напряжения питания процессор должен убедиться в правильности информации, сохраненной в статическом ОЗУ. Наиболее естественный путь — подсчет контрольной суммы и сравнение ее с эталоном. Однако есть проблемы с получением эталона. Более подробно этот вопрос рассмотрен в § 6.14.

2.2. Арбитры, реализующие гибкое обслуживание запросов

Как отмечалось, одновременные обращения нескольких устройств к магистрали приводят к конфликтам, которые разрешаются арбитром. Арбитр, показанный на рис. 2.2, имеет «жесткую» логическую структуру. Ему свойственно «оттеснение» высокоприоритетными каналами низкоприоритетных, в результате последние могут полностью лишаться права доступа к магистрали в течение длительных интервалов времени, что не всегда допустимо.

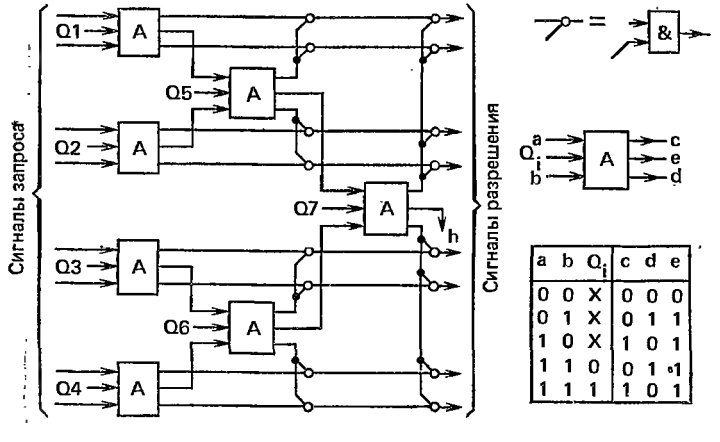
Одним из возможных путей повышения гибкости арбитра является периодическая программная перестройка его приоритетной структуры. Операционная система, например, через каждую миллисекунду меняет местами приоритеты каналов по определенному алгоритму. Поэтому низкоприоритетные каналы периодически становятся высокоприоритетными, и наоборот. При хаотическом «перемешивании» приоритетов каналов все они оказываются в равных условиях. При целенаправленном перемешивании приоритетность в целом сохраняется, но эффекта «грубого оттеснения» нет.

На рис. 2.11 представлена схема арбитра, реализующего 128 различных приоритетных соотношений между восьмью входными каналами в зависимости от кода на его управляющих входах Q1—Q7 [18]. Каждому элементарному арбитру A соответствует один разряд Q_i входного программно-доступного регистра приоритетов (не показан). При отсутствии конфликтов элементарный арбитр транслирует сигналы a и b на выходы c и d без изменения. При конфликте элементарный арбитр «руководствуется указаниями» управляющего разряда Q_i. Можно убедиться, что при любой конфликтной ситуации на входах арбитра независимо от кода в регистре приоритетов на выходы пройдет только один сигнал. Перестройка приоритетной структуры достигается изменением кода на входах Q1—Q7.

Динамическая программная перестройка приоритетов решает проблему «оттеснения» низкоприоритетных каналов, однако с увеличением частоты перестройки уменьшение жесткости арбитра сопровождается увеличением доли машинного времени, теряемого, с точки зрения пользователя, бесполезно. Поэтому управление арбитром (рис. 2.11) может осуществляться от аппаратного генератора кон-

стант, построенного, например, на основе ПЗУ или ОЗУ. Автоматически сменяемые константы подаются на входы Q1—Q7 и воздействуют на приоритетную структуру арбитра.

На рис. 2.12 показана схема вероятностного арбитра, который не требует постоянной опеки со стороны операционной системы [19]. В каждом канале, требующем обслуживания, с помощью генератора случайных чисел вырабатывается число из диапазона, определяемого содержимым программно-доступного регистра диапазона.



a	b	Q_i	c	d	e
0	0	X	0	0	0
0	1	X	0	1	1
1	0	X	1	0	1
1	1	0	0	1	0
1	1	1	1	0	1

Рис. 2.11. Арбитр с программируемыми приоритетами каналов (h — выход расширения)

Из полученной группы случайных чисел блок выбора максимального числа выделяет наибольшее и переводит в единичное состояние соответствующий триггер выходного регистра (при равных числах побеждает верхний канал). Сигнал разрешения доступа проходит через логический элемент ИЛИ и приостанавливает работу арбитра до снятия обслуживаемого запроса, после чего сбрасываются соответствующие триггеры входного и выходного регистров. Далее цикл работы повторяется. Управление приоритетами достигается путем записи констант в регистры диапазона. Иными словами, схема (рис. 2.12) осуществляет «игру в кости» между каналами, требующими обслуживания. Чаще (но не всегда) выигрывает тот канал, у которого «кость» имеет большее число граней.

Более простая по аппаратной реализации (но и не совсем «вероятностная») схема, предложенная в [20], показана на рис. 2.13. Каждый запросный сигнал встречается со всеми возможными конкурентами в одной из узловых схем. Из таблицы истинности, описывающей поведение узловой схемы (рис. 2.13, *в*), следует, что при отсутствии элементарного конфликта узловая схема (рис. 2.13, *б*) просто транслирует входные сигналы на соответствующие выходы, а при наличии конфликта — «руководствуется указаниями» триггера Q. Все 28 управляющих триггеров (по числу узловых схем) могут яв-

ляться, например, разрядами одного программно-доступного сдвигового регистра с одним входом для подключения к какой-либо линии шины данных. Будем пока считать, что код в регистре не меняется. Арбитр устраняет любые конфликты при любом коде в регистре, однако в зависимости от кода он может быть настроен либо на один из жестких, либо на один из гибких режимов.

При отсутствии запросных сигналов сигналы разрешений также отсутствуют, т. е. на всех выходах арбитра сформированы сигналы

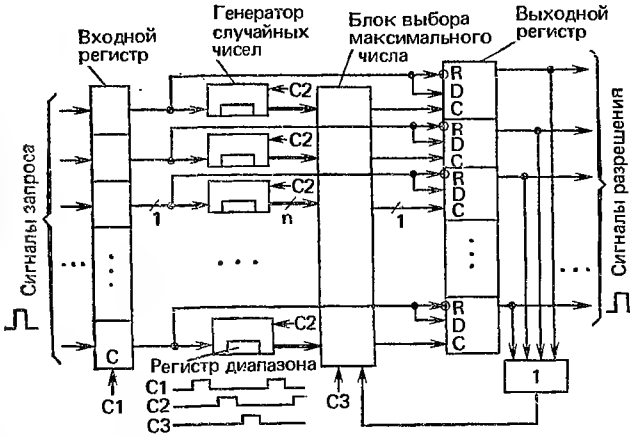


Рис. 2.12. Вероятностный арбитр (первый вариант)

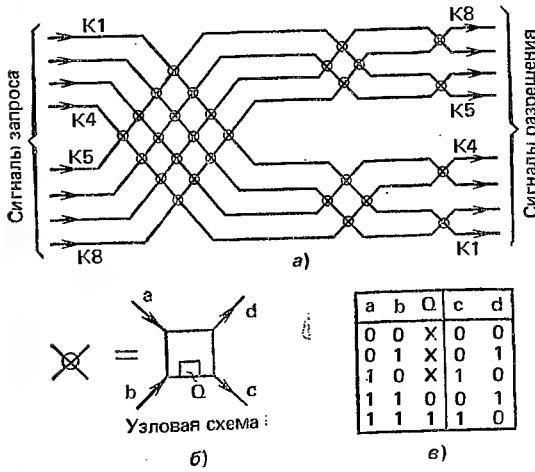


Рис. 2.13. Вероятностный арбитр (второй вариант)

лог 0 (см. верхнюю строку таблицы, рис. 2.13, *в*). При наличии одиночного запроса, представленного сигналом лог. 1 на одном из входов арбитра, этот сигнал проходит через все встречающиеся на его пути узловые схемы и поступает на соответствующий выход арбитра в виде сигнала лог. 1. Этот результат не зависит от кода в 28-разрядном управляющем регистре (см. символ × в строках 2 и 3 таблицы, рис. 2.13, *в*).

Если на входы арбитра одновременно подано несколько сигналов лог. 1, то эти сигналы, встречаясь каждый с каждым в соответствующих узлах решетки, подвергаются отбору, в результате на выходы арбитра попадает лишь один сигнал лог 1. Взаимоотношения между любой парой конкурирующих запросных сигналов однозначно определяются управляющим кодом в регистре (см. строки 4 и 5 таблицы рис. 2.13, *в*).

Пусть, например, требуется обеспечить следующую приоритетность входов К1—К8 (нумерация приведена на рис. 2.13):

К3→К2→К8→К6→К7→К1→К5→К4.

Стрелка и такой записи направлена от более приоритетного входа к менее приоритетному. Для обеспечения указанного режима управляющий код в регистре должен настроить узловые схемы следующим образом:

К3→К2, К3→К8, К3→К6, К3→К7, К3→К1, К3→К5, К3→К4*,
К2→К8, К2→К6, К2→К7, К2→К1, К2→К5, К2→К4,
К8→К6, К8→К7, К8→К1, К8→К5, К8→К4,
К6→К7, К6→К1, К6→К5, К6→К4,
К7→К1, К7→К5, К7→К4,
К1→К5, К1→К4,
К5→К4.

Каждое из этих условий задается значением соответствующего бита управляющего кода в 28-разрядном регистре. Аналогично можно задать любой жесткий приоритетный порядок между каналами. Число таких режимов равно числу перестановок между номерами каналов $P_8=40\,320$. Однако это число не исчерпывает все возможные режимы работы устройства. Действительно, поскольку число управляющих бит в арбитре в данном примере составляет $C_8^2=28$, то возможны $2^{28}=268\,435\,456$ вариантов настройки, которые, за исключением уже рассмотренных, характеризуются гибким распределением приоритетов между каналами.

Смысл этих режимов состоит в том, что приоритет между каналами определяется не только управляющим кодом в регистре, но также зависит от числа поступивших запросов и их распределения по каналам. Если в предыдущем примере изменить значение только одного бита в управляющем коде, а именно бита, определяющего взаимоотношения между входами К3 и К4, то условие К3→К4, помеченное ранее знаком *, изменится на противоположное: К4→К3 («шестерка бьет туза»). Такое изменение, как будет показано, приводит к полученню одного из гибких режимов.

Предположим, что мы имеем дело с той же приоритетной структурой, но с тем отличием, что К4→К3. Предположим далее, что запросные сигналы поступили по всем входам К1—К8. На рис. 2.14, *а* полужирными линиями показаны траектории распространения запросных сигналов через арбитр. Стрелка, направленная из узла, по-

казывает, какому сигналу отдается предпочтение при конфликте. Стрелка, направленная вправо и вверх, соответствует сброшенному состоянию соответствующего управляющего триггера Q; стрелка, направленная вправо и вниз, — единичному состоянию этого триггера (см. строки 4 и 5 таблицы, рис. 2.13, *в*). Из диаграммы, показанной на рис. 2.14, *а*, видно, что при взаимодействии сигналов К4 и К5 побеждает К5, так как соответствующая узловая схема решетчатой структуры настроена на приоритетную передачу сигнала, поступившего на вход К5 при возникновении конфликта между К5 и К4 (К5→К4).

Указанный сигнал далее взаимодействует с сигналом, поступившим на вход К3, и на этот раз прекращает существование, так как не выдерживает конкурентной борьбы с этим сигналом (К3→К5).

Рис. 2.14. Траектории распространения запросных сигналов через арбитр:

а — при наличии всех запросных сигналов; *б* — при наличии четырех запросных сигналов по каналам К1—К4

Результирующий сигнал с входа К3 последовательно «побеждает» сигналы, поступившие по входам К6, К7, К8, К2 и К1, и поскольку в узле, помеченном знаком *, уже нет конкурента по каналу К4 (К4→К3), беспрепятственно проходит на выход арбитра.

Сигнал, поступивший по входу К2, хотя и имеет более высокий приоритет по отношению к сигналам, поступившим с входов К5—К8, но не встречается с ними на своем пути, так как последние подавлены сигналом, пришедшим по входу К3; сигнал, поступивший по входу К2, пройдя до встречи с сигналом, поданным на вход К3, прекращает существование (К3→К2). Сигнал, поданный на вход К1, имеет сравнительно низкий приоритет (пять стрелок из семи стремятся «отклонить» сигнал от продвижения по каналу К1 при конфликтах), однако, не встречая на своем пути конкурентов, он достаточно глубоко проникает в структуру и прекращает существование, лишь встретившись с сигналом, распространяющимся по каналу К3.

Таким образом, при данной настройке узловых схем решетчатой структуры и при поступлении всех без исключения запросных сигналов на выход пройдет сигнал по каналу К3. Однако из этого нельзя сделать вывод о том, что канал К3 — самый «сильный». Действительно, покажем, что при поступлении запросов не по восьми, а, например, только по четырем входам К1—К4 выигрывает не К3, а К4.

Рассмотрим схему взаимодействия сигналов, приведенную на

рис. 2.14, *б*. Сигналы, поступившие по входам К1 и К2, встречаясь с сигналом, поданным на вход К3, выбывают из конкурентной борьбы. Сигнал, пришедший с входа К4, не встречая «сопротивления» со стороны сигналов по входам К5—К8, К2 и К1, проходит до узла, помеченного знаком *, и приостанавливает распространение сигнала, пришедшего по входу К3, так как по условию К4→К3. Таким образом, при числе запросов, равном четырем, и их распределении по каналам согласно рис. 2.14, *б* канал К4 имеет максимальный приоритет.

Здесь наблюдается парадоксальная ситуация: канал К3 одерживает победу над всеми одновременно действующими конкурентами, но проигрывает некоторым их подмножествам, например единственному конкуренту — каналу К4. Заметим, что этот результат получен изменением лишь одного бита в управляющем коде. При изменении большего числа бит мы будем все более удаляться от структуры с жесткими приоритетами и придем к потрясающей воображение путанице приоритетов, в которой, однако, легко разобраться с использованием программной модели арбитра.

Таким образом, любой гибкий режим в той или иной мере снижает жесткость арбитра. Приоритеты каналов уже не являются абсолютными, а имеют вероятностный характер, по крайней мере при не слишком высокой активности каналов. С повышением их активности режим обслуживания становится все менее демократичным. В частности, при постоянном воздействии на входы арбитра всех без исключения запросных сигналов побеждает всегда один и тот же канал. (Может быть, это не так и плохо в некоторых применениях?) Более подробная информация об этой и других проблемах приведена в § 7.19.

2.3. Типы контроллеров

На рис. 2.15 показана типовая схема связи микроЭВМ с объектом. Таким объектом может являться стандартное устройство ввода-вывода (дисплей, накопитель на магнитных дисках и т. д.) или нестандартное (система датчиков и исполнительных органов технологической установки, цеховая система сбора информации с рабочих мест и т. д). Как отмечалось, возможности обмена процессора микроЭВМ с контроллером исчерпываются всего лишь тремя режимами: режимом чтения информации из программно-доступных элементов контроллера (регистров, отдельных внутренних точек, триггеров и др.) как из ячеек памяти; режимом записи информации в программно-доступные элементы контроллера как в ячейки памяти; режимом прерывания основной (фоновой) программы по сигналу из контроллера с переходом на подпрограмму обслуживания данного контроллера. (Процессор умеет читать, писать и прерываться, только и всего!) Этот факт в значительной степени упрощает подход к проектированию контроллеров.

На рис. 2.16 приведена классификация контроллеров согласно [21]. Синхронный обмен используется редко и только для процессов, строго фиксированных во времени и полностью определенных. Внешнее устройство должно быть готово для осуществления обмена. Программа работы микроЭВМ при синхронном обмене проста и понятна (рис. 2.17, *а*). Примерами устройств, с которыми может осуществляться синхронный обмен, являются: светодиодный индикатор, обмот-

ка реле, датчик случайных чисел на неоновой лампочке и другие «всегда готовые» устройства, в основном с низким интеллектом.

Асинхронный обмен (рис. 2.17, *б*) используется очень часто. Он осуществляется под управлением программы, но лишь в том случае, когда внешнее устройство находится в состоянии готовности. Команда проверки готовности используется для ввода в микроЭВМ информации о состоянии внешнего устройства. Затем процессор производит анализ состояния и принимает решение. Если устройство не готово, то процессор возвращается к команде проверки готовности и т. д. Когда устройство перейдет в состояние готовности, процессор приступает к обмену. Основным недостатком асинхронного обмена являются потери времени на ожидание того момента, когда устройство окажется готовым к обмену.

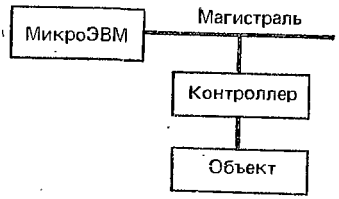


Рис. 2.15. Типовая схема связи микроЭВМ с объектом

Обмен по прерыванию позволяет более эффективно использовать машинное время. Обмен в этом случае осуществляется под управлением программы, но микроЭВМ многократно не проверяет готовность внешнего устройства до момента ее установления. МикроЭВМ может выполнять работу, не связанную с данным обменом (фоновую программу). Когда внешнее устройство готово для обмена, оно формирует сигнал прерывания, вызывающий приостановку фоновой программы и переход к выполнению подпрограммы обработки прерывания. Эти действия схематически показаны на рис. 2.17, *в*. После выполнения подпрограммы управление вновь передается фоновой программе.

В [21] в качестве примера проводится сравнение синхронного, асинхронного и обмена по прерыванию, когда одному из членов семьи поручено следить за кипячением молока и снять его с плиты, как только оно будет готово. Безусловная операция: пойти на кухню, скажем, в 8 ч 25 мин и снять молоко с плиты, не заботясь о том, вскипело оно или нет. Операция по условию: присматривать за молоком, скажем, раз в минуту. Когда оно закипит, снять его с плиты. Опера-

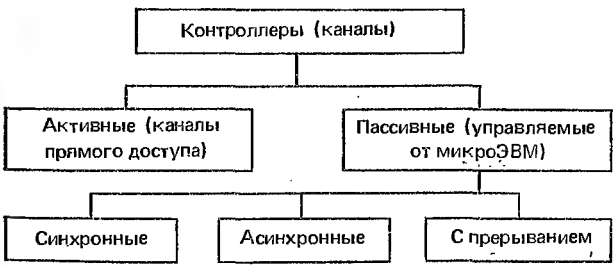


Рис. 2.16. Классификация контроллеров

ция по прерыванию: установить сигнальное устройство, которое звонит при закипании молока. Выполнять фоновую домашнюю работу. Когда зазвенит звонок, прервать фоновую работу и снять молоко с плиты. Возобновить фоновую работу с того места, на котором она была прервана.

Канал прямого доступа представляет собой «интеллектуальное» устройство, способное самостоятельно, без помощи центрального процессора, вводить информацию в память микроЭВМ или выводить ее

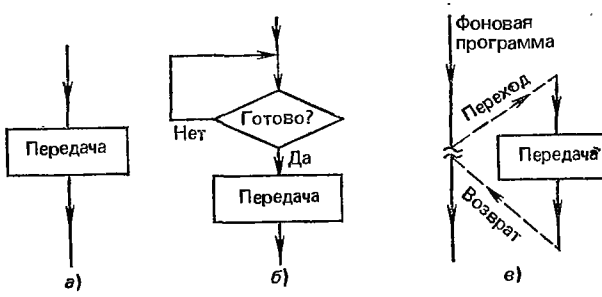


Рис. 2.17. Алгоритмы работы контроллеров:
а — синхронного; *б* — асинхронного; *в* — с прерыванием

из памяти. Центральный процессор может участвовать лишь в постановке задачи каналу. После выдачи задания он уже не принимает участия в передаче и, если есть такая возможность, продолжает работу по собственной программе. Канал прямого доступа по окончании передачи массива информирует об этом процессор, например, посылкой ему сигнала прерывания.

2.4. Примеры пассивных контроллеров

На рис. 2.18 показан пример схемы синхронного контроллера. Предполагается, что в адресном пространстве заняты первые и последние 8К адресов. Интерфейс — «общая шина». Контроллер предназначен для сопряжения микроЭВМ со световым табло. Предполагается, что машина по собственной инициативе меняет информацию на световом табло, например, выдавая рекламные сообщения. Для смены информации микроЭВМ сначала последовательно загружает сдвиговый регистр RG1, а затем переписывает его содержимое в параллельный регистр RG2 для того, чтобы кадр сменился мгновенно, без хаотических переходных процессов.

Для загрузки регистра RG1 микроЭВМ N раз выполняет команду записи типа регистр — память. В качестве адреса используется любой адрес из диапазона 8000—BFFF (реализован метод частичной дешифрации). Для переписи содержимого регистра RG1 в регистр RG2 процессор микроЭВМ обращается в режиме чтения или записи по одному из адресов в пределах 4000—7FFF, выполнив, например, команду пересылки слова из регистра R0 процессора в ячейку памяти с адресом 7777 : R0→ [7777]. Распределение адресного пространства показано на рис. 2.19.

Элемент ИЛИ — НЕ оповещает микроЭВМ о завершении обмена. Однако его можно исключить из схемы. При этом ответный сигнал не будет вырабатываться и микроЭВМ воспримет эту ситуацию как ошибочную. Поэтому примерно через 10 мкс процессор автоматически перейдет в режим прерывания, т. е. запомнит свое текущее состояние и перейдет к программе, адрес начала которой находится в ячейке X ОЗУ (адрес ячейки X должен быть указан в эксплуатационной документации на микроЭВМ). Так как это зависание является «запланированным» (мы знали, на что идем), то на него не надо реагировать как на ошибку. Поэтому непосредственно перед обращением

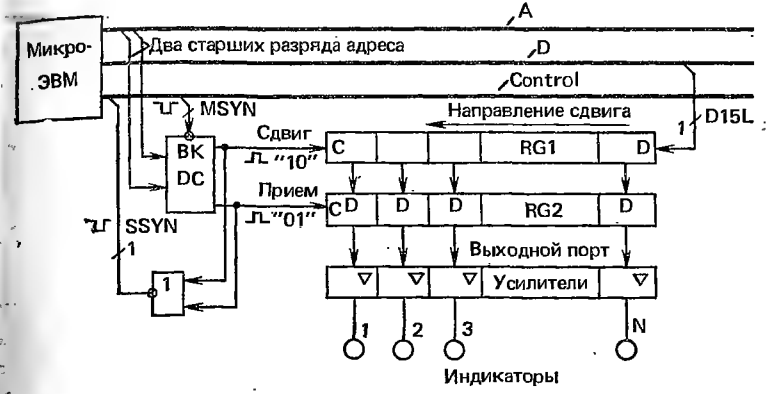


Рис. 2.18. Синхронный контроллер

к контроллеру «нормальная» программа реакции на зависание как на ошибку подменяется фиктивной, состоящей из единственной команды *выход из прерывания*. После обращения к контроллеру «нормальная» программа должна быть восстановлена. Подмена программ производится простым изменением содержимого ячейки X.

Регистр RG1 можно было бы загружать не по одному разряду, а группами по 16 разрядов. Перепись из регистра RG1 в регистр RG2 можно было бы производить без участия микроЭВМ, определив с помощью счетной схемы момент готовности информации в регистре RG1. Нужны ли подобные улучшения? Если требования к скорости смены кадров на табло удовлетворяются и без этих улучшений, то они, конечно, не нужны. При проектировании контроллера старайтесь переложить как можно больше функций на микропроцессор. Под управлением программы он может формировать на адресной шине любые желаемые адреса, а их дешифрация контроллером порождает формирование нужной последовательности управляющих импульсов в вашей схеме.

На рис. 2.20 показана схема асинхронного контроллера (под контроллером на рис. 2.20 подразумевается «всё, кроме микроЭВМ»). Задача состоит в том, чтобы выдавать на табло оценки спортивных судей. МикроЭВМ периодически проверяет готовность всех судей к выдаче оценок. Когда появляются все сигналы готовности, микро-

ЭВМ принимает оценки, обрабатывает их, вычисляя, например, средний балл, и выдает информацию на табло. Таким образом, условием для ввода и вывода является наличие лог. 1 (сигиалов готовности) на определенных входах. Распределение адресиого пространства приведено на рис. 2.21.

Для чтения информации от объекта микроЭВМ должна открыть логическне элементы И, в результате информация параллельным кодом запишется в регистр RG3, а затем произвести серию сдвигов. При каждом сдвиге очередной бнт поступает в микроЭВМ по линии D08L (лнния выбрана произвольно). Процессор микроЭВМ собирает эти биты в 16-разрядные слова программным путем. Для приема информации в регистр RG3 процессор мнкроЭВМ обращается в режиме чтения или записи по одному из адресов C000—DFFF, выполнив, например, команду пересылки слова из регистра R0 процессора в ячейку памяти с адресом DDD0 : R0→[DDD0]. При этом срабатывает дешифратор, открываются логические элементы И и в микроЭВМ через логический элемент ИЛИ—НЕ посылается ответный сигнал SSYN.

Рис. 2.19. Распределение адресного пространства микроЭВМ с синхронным контроллером

При сдвиге процессор микроЭВМ выполняет операцию чтения, например, из ячейки 2ABC: [2ABC]→R0. После выполнения этой команды в разряде 8 регистра R0 окажется очередной бит, выдвинутый из регистра RG3. Программа собирает из этих битов слово, относящееся к полю готовности, и проверяет его на наличие едниц. Если нет хотя бы одной единицы, то дальнейший ввод прекращается и микроЭВМ вновь начинает процедуру с загрузки регистра RG3. Если все поле готовности заполнено единицами, то ввод продолжается, в результате информация об оценках попадает в память мнкроЭВМ. Элемент задержки Δt гарантирует выдачу сигнала SSYN не раньше, чем закончатся переходные процессоры в регистре RG3 при сдвиге.

На рис. 2.22 показана схема контроллера с прерыванием. В состав этой схемы включена схема контроллера, показанная на рис. 2.20 с тем отличием, что входы готовности теперь соеднпены с входами логнческого элемента И.

В исходном состоянии микроЭВМ выполняет фоновую задачу и «не обращает внмаиия» на контроллер. При наличии общей готовности срабатывает логический элемент И, запуская схему захвата магистрали. Захватив магистраль, контроллер выдаёт через логические элементы И вектор прерывания, который принимается процессором. Программа, вызываемаи по вектору, «знает», что сигналы готовности имеются (иначе она бы не вызвалась). Поэтому она считывает оценки и выдает иа табло нужную информацию уже без проверки

каких-либо условий. После этого микроЭВМ возвращается к фоновой задаче.

Схема захвата магистрали приведена на рис. 2.23. Для определенности выбрана линия запроса BR5 и соответственно линия разрешения BG5. Алгоритм захвата магистрали рассмотрен в § 2.1. В исходном состоянии все триггеры установлены в 0. Триггер запроса

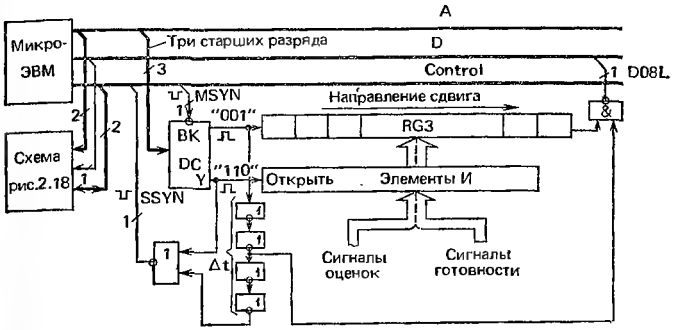


Рис. 2.20. Асинхронный контроллер

ТгЗпр устанавливается в единичное состояние при воздействии положительного фронта сигнала на его вход С. Триггер подтверждения запроса ТгПтЗ устанавливается в единичное состояние при наличии запроса (триггер ТгЗпр установлен в единичное состояние) в момент поступления сигнала разрешения BG5 за счет срабатывания логического элемента И1. Триггер ТгЗпр сбрасывается сразу после установки

Рис. 2.21. Распределение адресного пространства микроЭВМ с асинхронным контроллером

в единицу триггера ТгПтЗ. Установив в единичное состояние триггер ТгПтЗ, контроллер «знает», что именно он является первоочередным претендентом на занятие магистрали. Теперь для занятия магистрали нужно дождаться ее освобождения от текущего обмена, если он есть. В момент освобождения срабатывает логический элемент И2 и устанавливается в единичное состояние триггер собственной занятости ТгЗанс. Триггер ТгПтЗ при этом устанавливается в 0.

После этого контроллер по праву считает себя «владельцем» магистрали. В момент занятия магистрали по входу С устанавливается в единичное состояние триггер прерывания ТгПрер, в результате открываются элементы И выдачи вектора в шину данных. Процессор микроЭВМ принимает вектор и отвечает контроллеру сигналом SSYN, начало которого устаиавливает в 0 триггер прерывания. По окончании сигнала SSYN устанавливается в 0 триггер ТгЗанс,

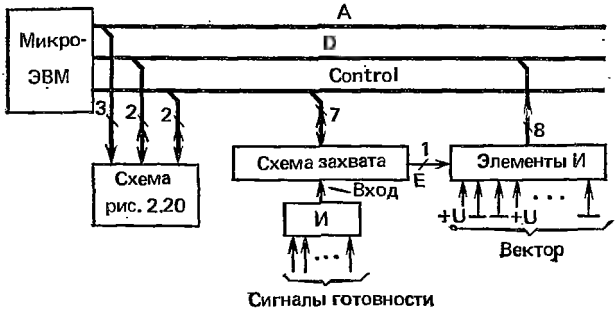


Рис. 2.22. Контроллер с прерыванием

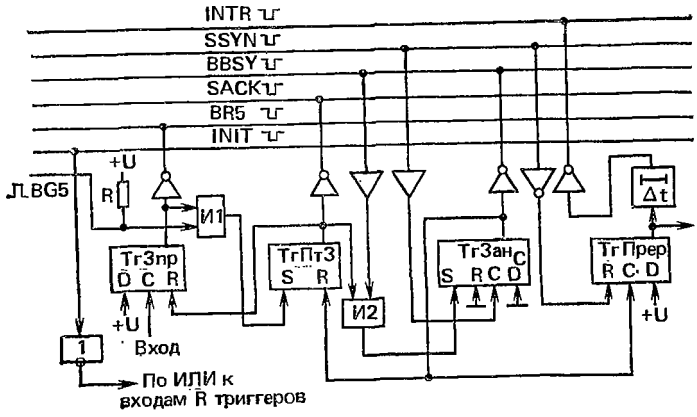


Рис. 2.23. Схема захвата магистрали

после чего магистраль освобождается, а контроллер переходит в исходное состояние.

Сигнал с интерфейсной линии INIT («общий сброс») должен быть «смешан» по схеме ИЛИ с сигналами функциональных сбросов триггеров, чтобы в необходимых случаях, например после включения напряжения питания, схема приводилась в исходное состояние. Элементы ИЛИ на рис. 2.23 не показаны для его упрощения.

Как отмечалось в § 2.1, если контроллер не является последним в группе, то при отсутствии «нашего» запроса необходимо транслировать сигнал разрешения BGi с входа контроллера на его выход. Поскольку моменты формирования запросов активными устройствами заранее не известны, не исключена возможность появления «нашего» запроса в момент прохождения сигнала разрешения через «наше» активное устройство к «чужому» активному устройству, распо-

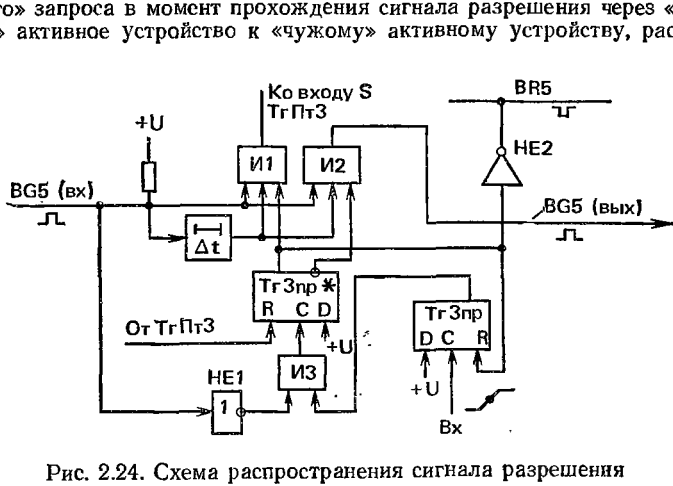


Рис. 2.24. Схема распространения сигнала разрешения

ложенному справа (по схеме рис. 2.2), в ответ на его запрос. Если не принять специальных мер, то может оказаться, что, хотя «наше» устройство и «перехватило» сигнал разрешения, на выходе все же успеет появиться кратковременный сигнал. Если этот кратковременный сигнал воспримется одним из последующих устройств, то это устройство захватит магистраль одновременно с «нашим», что недопустимо. Такие ситуации могут возникать очень редко, например один раз в сутки, — тем более трудно их зарегистрировать и разобраться в причинах их появления. Поэтому схемы асинхронного взаимодействия сигналов требуют тщательного критического анализа на уровне «мысленного эксперимента» в процессе их разработки.

На рис. 2.24 приведен фрагмент схемы захвата магистрали, снабженной средствами передачи сигнала разрешения по цепочке контроллеров [22]. При поступлении сигнала разрешения закрывается логический элемент И3, запрещая установку в 1 триггера ТгЗпр* в период прохождения сигнала разрешения. Через время, превышающее сумму задержек инвертора НЕ1, логического элемента И3 и триггера ТгЗпр*, триггер ТгЗпр* окажется в заведомо устойчивом состоянии (0 или 1, но не 0→1). Поэтому либо устанавливается в 1 триггер

ТгПтЗ, либо разрешенне поступает в следующий контроллер. Другие (некорректные) варианты исключены. При окончании сигнала входного разрешения логические элементы И1 и И2 закрываются по входам, а схема вновь становится чувствительной к сигналу запроса.

Вернемся вновь к схеме контроллера, приведеиной на рис. 2.22, и рассмотрим ее с более общих позиций как одну из возможных схем контроллера программируемого интерфейса (КПИ). Смысл этого названия будет ясен из последующих примеров использоваиия КПИ, а пока отметим главные его свойства.

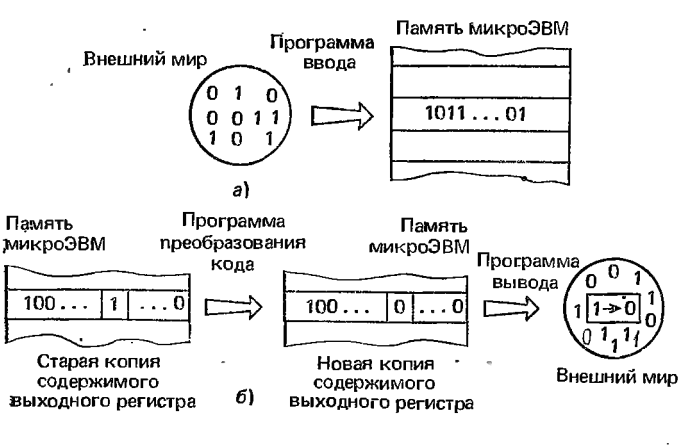


Рис. 2.25. Программная поддержка КПИ:

а — при вводе информации; *б* — при выводе информации

1. Контроллер программируемого интерфейса позволяет по инициативе программы прочитать информацию с внешних входов и разместить ее в памяти микроЭВМ. После этого информация может подвергнуться самым различным видам обработки: можно, например, выделять отдельные поля, проверять, изменять отдельные биты, выполнять арифметические и логические операции и т. д. Чтобы каждый раз не вспоминать о том, что ввод информации с входов контроллера программируемого интерфейса сопровождается загрузкой регистра RG3 (см. рис. 2.20), серией сдвигов и программной упаковкой отдельных бит, будем использовать в последующих примерах язык более высокого уровня. Так, выражение *прочитать информацию с входов КПИ* соответствует выполнению десятков или сотен команд программы ввода (рис. 2.25, *а*).

2. Контроллер программируемого интерфейса позволяет по инициативе программы менять код в выходном регистре, причем новый код может отличаться от старого только в одном разряде или в группе разрядов, а смена кода производится «мгновенно» и не затрагивает те биты, которые не меняют состояния. Поэтому с помощью выходного регистра КПИ можно одновременно управлять несколькими независимыми объектами, например печатающим устройством и цифро-

аналоговым преобразователем. Чтобы изменнть информацию на выходах контроллера (сразу на всех выходах или только на части выходов без измеиения оставшихся), нужно программно подготовить в памяти микроЭВМ новую копию содержимого выходного регистра контроллера и переслать ее в этот регистр. Выдача информации в выходной регистр контроллера (см. рис. 2.18) может сопровождаться выполнением десятков или сотен команд распаковки, сдвига, подсчета числа циклов и т. д. Чтобы каждый раз не напоминать об этих «подробностях», будем использовать в дальнейшем выражения типа *изменить бит 60 в выходном регистре КПИ*.

3. Контроллер программируемого интерфейса позволяет оповещать процессор о событиях во внешнем мире с помощью прерываний. Этот вопрос уже достаточно подробно рассмотрен.

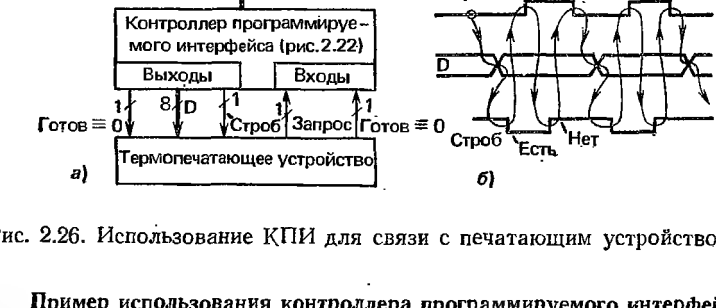


Рис. 2.26. Использование КПИ для связи с печатающим устройством

Пример использования контроллера программируемого интерфейса для связи микроЭВМ с термопечатающим устройством. На рис. 2.26 показана схема связи и временна́я диаграмма обмена. Используется версия стандартного интерфейса ИРПР. Низкий уровень напряжения (лог. 0) на линии Запрос означает, что термопечатающее устройство готово принять очередной байт по линиям данных D. Истинность выдаваемого из контроллера байта подтверждается отрицательным фронтом сигнала на линии Строб. В байте закодирован символ либо команда перевода строки. Распределение разрядов на выходах и входах контроллера (называемых обычно выходным и входным портами) — произвольное. Необходимыми условиями осуществления сеанса связи являются низкие уровни напряжения на линиях готовности (Готов) с обеих сторон. Опишем подпрограмму вывода информации на печать.

1. Установить в 0 разряд Готов КПИ.
2. Прочитать информацию с входов КПИ.
3. Если на обоих входах — нули, перейти к п. 4, в противном случае — к п. 2.
4. Выдать очередной байт на выходы D (символ либо команду перевода строки).
5. Выдать 0 в разряд Строб.
6. Прочитать информацию с входов КПИ.
7. Если Запрос=1, то перейти к п. 8, иначе — к п. 6.
8. Выдать 1 в разряд Строб.

9. Если все байты выданы, перейти к п. 10, иначе — к п. 2.
10. Выйти из подпрограммы.

Примечание. Во избежание зацикливания между пп. 3 и 2, 7 и 6 можно ввести команды подсчета этих циклов и сравнения с максимально допустимым числом. При превышении этого числа, например, из-за потери готовности термопечатающего устройства при отсутствии бумаги производится выход из подпрограммы.

Пример использования контроллера программируемого интерфейса для связи с аналоговым датчиком. На рис. 2.27 показано подключение цифро-аналогового преобразователя и аналогового компаратора к выходному и входному портам КПИ. Опишем программу измерения значения аналогового сигнала y.

Рис. 2.27. Использование КПИ для связи с аналоговым датчиком

Рис. 2.28. Использование КПИ для связи между двумя микроЭВМ

1. Установить в 0 выходы D.
2. Программная задержка 100 мкс.
3. Прочитать информацию с входа z.
4. Если $\dot{z}=0$, перейти к п. 5, иначе — к п. 7.
5. Прибавить единицу младшего разряда к копии кода D.
6. Переслать копию из памяти в выходной порт и перейти к п. 2.
7. Выход из подпрограммы.

Копия кода D представляет собой кодированное значение аналоговой величины y. В данном примере не ставилась задача минимизации времени измерения.

Пример использования контроллеров программируемого интерфейса для связи между двумя микроЭВМ. На рис. 2.28 показана схема связи двух микроЭВМ с помощью двух КПИ. Промежуточный интерфейс — ИРПР (см. рис. 2.26, *б*). Предположим, что микроЭВМ1 подготовила в своей памяти массив информации для передачи в микроЭВМ2 и послала сигнал уведомления об этом на вход схемы захвата магистрали КПИ2 через один из разрядов выходного регистра КПИ1. МикроЭВМ2 откладывает текущее задание и переходит к подпрограмме связи с микроЭВМ1. При этом с точки зрения микроЭВМ1, микроЭВМ2 ведет себя аналогично термопечатающему устройству (см. рис. 2.26). Считанная через КПИ2 информация раз-

мещается в памяти микроЭВМ2. Признаком конца передачи может являться выдача единицы на одну из связей Y. По окончании сеанса связи обе микроЭВМ возвращаются к выполнению своих текущих заданий. Передача массива из микроЭВМ2 в микроЭВМ1 может производиться аналогично.

Пример использования контроллера программируемого интерфейса для подключения к микроЭВМ дополнительного ОЗУ (рис. 2.29). Входы-выходы данных ОЗУ соединены с выходным и входным портами КПИ. Интерфейс—«общая шина» (см. § 2.1). Ответный сигнал от ОЗУ (SSYN) не анализируется—он заведомо имеется к моменту опроса процессором входного порта КПИ (при чтении) или к моменту снятия сигналов A, D, C1, MSYN (при записи). Программное воплощение временных диаграмм (см. рис. 2.4, 2.5) приво-

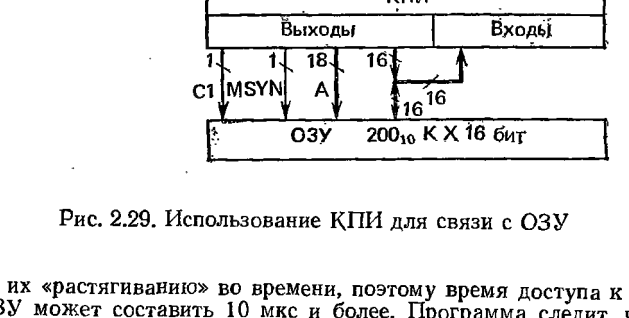


Рис. 2.29. Использование КПИ для связи с ОЗУ

дит к их «растягиванию» во времени, поэтому время доступа к ячейке ОЗУ может составить 10 мкс и более. Программа следит, чтобы адрес ие превышал 200К—1. Схема (рис. 2.29) может использоваться в качестве стеида для поиска неисправностей в отказавших блоках ОЗУ.

Пример использоваиия КПИ для загрузки внешнего сдвигового регистра (рис. 2.30). Для правильиой работы сдвигового регистра необходимо в каждом из 1000 программиых циклов работы микроЭВМ подать на его вход D очередной информационный бит, а затем на управляющий вход С — положительный фронт сигнала. После загрузки всех 1000 разрядов на одном из выходов КПИ программно формируется импульс, подтверждающий факт загрузки регистра. (По этому импульсу можно переписать информацию из сдвигового регистра в дополнительный параллельный регистр (не показан), как это делалось в схеме рис. 2.18; тем самым можно нарастить разрядность выходного порта КПИ.) Пример использования подобной схемы для сбора информации от 1024 проверяемых в режиме прогона микроЭВМ описаи в § 5.24. Отметим, что для правильной работы цепи из микросхем сдвиговых регистров необходимо, чтобы сигнал С, проходя по цепи ретрансляторов, распространялся навстречу сдвигаемой информации, иначе могут возникнуть «гонки» и информация при сдвиге будет искажаться.

Пример использования КПИ для работы с нестандартным последовательным каналом связи (рис. 2.31 . Истинность принимаемой

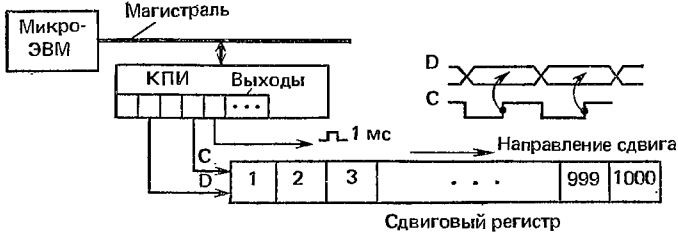


Рис. 2.30. Использование КПИ для загрузки внешиего сдвигового регистра

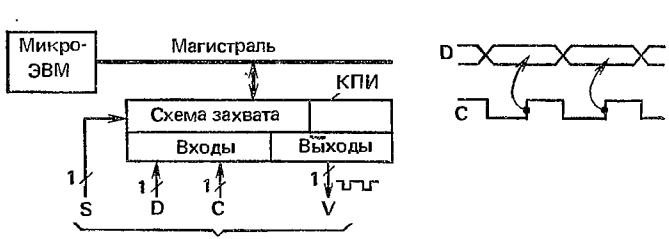


Рис. 2.31. Использование КПИ для работы с последовательным каналом связи

по линии D информации гарантируется в промежутке времени длительностью не менее 100 мкс с момента появления положительного фронта сигнала на входе C. Программа приема информации из канала вызывается по прерыванию сигналом S начала сеанса связи; этот сигнал поступает из канала за 1—2 мс до начала передачи данных. По окончании приема информации на выходе V КПИ программно формируются два отрицательных (по фазе) импульса длительностью 1 мс с паузой 5 мс. Длина принимаемого сообщения 70К бит.

Контроллер программируемого интерфейса может содержать программно-доступный N-разрядный управляющий регистр. В зависимости от кода в этом регистре каждая из N внешних линий контроллера может быть назначена либо входом, либо выходом. После включения напряжения питания все N линий автоматически назначаются входными, для чего можно использовать сигнал на линии ACLO (см. § 2.1).

Приведенные примеры использования КПИ, конечно, не исчерпывают все возможности его применения, а лишь показывают его универсальность. Любой интерфейс, разработанный вами для взаимодействия микроЭВМ с объектом, может быть реализован программно с помощью КПИ, если, конечно, КПИ имеет достаточное число входных и выходных линий, а микроЭВМ обладает нужным быстродействием. Проще говоря, с помощью КПИ вы можете «запрограммировать» любую временную диаграмму, изображенную на листе бумаги

(см. например, рис. 2.26, *б*), если нет жестких ограничений на время между фронтами сигналов. Если, например, время между положительными фронтами сигналов Запрос и Строб не должно было бы превышать 100 нс, то соответствующий фрагмент временной диаграммы пришлось бы реализовать схемно, так как микроЭВМ не успеет смоделировать его столь быстро. Схема в этом случае должна передавать сигнал с линии Запрос непосредственно в линию Строб, не ожидая «помощи» от процессора (программы).

2.5. Методы поиска источников прерывания

При наличии многих источников прерывания вычислительной машине нужно разобраться, какой из них прислал сигнал прерывания, для того чтобы перейти к соответствующей программе обслуживания. В интерфейсе «общая шина» каждому источнику прерывания может быть присвоен отдельный вектор. Однако такой путь не является единственно возможным. Более того, далеко не все микроЭВМ способны «понимать» векторные прерывания. Многие микроЭВМ содержат один или несколько входов прерываний, для каждого из которых в процессоре зафиксирован свой вектор. Поэтому при наличии, например, 50 источников прерывания их придется объединить в одну или несколько групп и вычислительная машина, в лучшем случае, может определить, в какой группе находится источник. Для выявления источника вычислительной машине нужна дополнительная информация. Рассмотрим три метода поиска источников прерывания в порядке возрастания степени аппаратной поддержки.

Метод программного последовательного опроса (рис. 2.32). МикроЭВМ, получив сигнал прерывания, «не знает», какой из контроллеров К1, К2, .., K_N его прислал, поскольку все запросные сигналы от контроллеров суммируются на общей линии по схеме Монтажное ИЛИ. (В интерфейсе «общая шина» такой линией может являться линия ACLO аварии питания. То же относится и к интерфейсу микроЭВМ «Электроника-60», где имеется линия ПИТ. В данном случае предлагается использовать эти линии не по прямому назначению. Поэтому при возникновении «настоящей» аварии питания (при выключении напряжения 220 В, 50 Гц) поведение процессора окажется не соответствующим реальной ситуации.)

После получения обобщенного сигнала прерывания микроЭВМ пытается выявить его источник путем последовательного опроса всех контроллеров как ячеек памяти. МикроЭВМ может сразу приступить

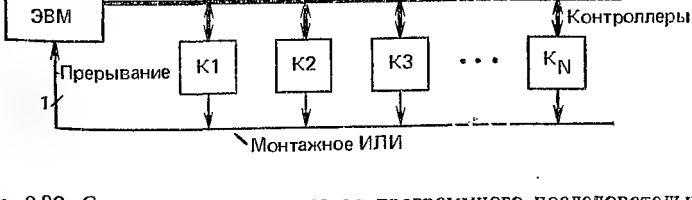


Рис. 2.32. Схема, поясняющая метод программного последовательного опроса

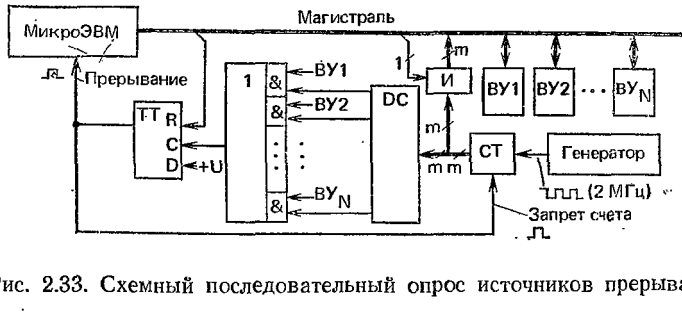


Рис. 2.33. Схемный последовательный опрос источников прерываний

к обслуживанию первого же обнаруженного контроллера, требующего обслуживания, либо оценить «степень бедствия», опросив все без исключения контроллеры и только после этого выбрать наиболее важный запрос. Приоритеты контроллеров в первом случае задаются последовательностью опроса, во втором — таблицей, хранимой в памяти. В любом случае недостаток метода — большое время реакции на прерывание. В худшем случае может оказаться, что обслуживания требовал контроллер, который опрашивался последним.

Метод схемного последовательного опроса (рис. 2.33, по материалам [23]). В исходном состоянии триггер ТТ сброшен, запросов от внешних устройств ВУ1—ВУ$_N$ нет, счетчик СТ через дешифратор DC последовательно циклически опрашивает запросные линии от внешних устройств, логические элементы И закрыты, сигнал сброса триггера отсутствует. При появлении одного или нескольких запросов срабатывает одна из схем И логического элемента И — ИЛИ, в результате триггер устанавливается в 1 и счетчик прекращает подсчет импульсов с генератора. При этом в счетчике фиксируется номер одного из внешних устройств, требующих обслуживания.

Единичный сигнал с выхода триггера вызывает прерывание микроЭВМ. Прерывающая программа открывает программно-доступные элементы И, в результате код из счетчика попадает в процессор и используется им для формирования адреса перехода к программе обслуживания выбранного внешнего устройства. Далее микроЭВМ заставляет выбранное внешнее устройство сбросить запрос и обслуживает его. Сброс триггера производится программно — после или в процессе обслуживания (в последнем случае возможны вложенные прерывания). После сброса триггера циклический опрос продолжается. На схеме, приведенной на рис. 2.33, не приведен ряд вспомогательных элементов: схемы стробирования дешифратора, магистральные приемопередатчики и т. п.

Метод схемного последовательного опроса позволяет значительно уменьшать время реакции на прерывание, так как опрос ведется с достаточно высокой скоростью. Аппаратные затраты не слишком велики. Основным недостатком является наличие радиальных связей логического элемента И — ИЛИ с внешними устройствами.

Метод схемного параллельного опроса (рис. 2.34), по материалам [24]). При появлении нескольких конкурирующих запросов на обслуживание или всего одного запроса микроЭВМ посылает одновременно во все внешние устройства ВУ1—ВУ$_N$ сигнал разрешения. По этому сигналу все заинтересованные в обслуживании внешние

устройства коллективно решают, кто из них «главнее». «Выяснение отношений» между этими устройствами идет без участия микроЭВМ с весьма высокой скоростью срабатывания распределенной по магистрали комбинационной схемы. Этот процесс завершается установлением на шине данных кода, который считывается машиной и рассматривается ею как вектор прерывания. Каждому внешнему устройству поставлен в соответствие свой вектор, он же задает приоритет устройства.

Все запросы от внешних устройств объединяются по схеме Монтажное ИЛИ на общей линии запросов BR. Регистр приоритета RG хранит код, определяющий важность запроса от данного внешнего устройства. Этот же код, как отмечалось, выполняет роль вектора прерывания процессора. Все коды приоритетов в разных внешних уст-

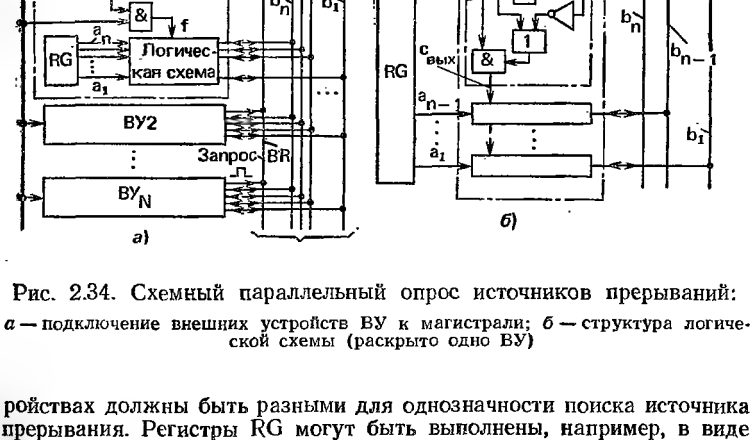


Рис. 2.34. Схемный параллельный опрос источников прерываний:
а — подключение внешних устройств ВУ к магистрали; *б* — структура логической схемы (раскрыто одно ВУ)

ройствах должны быть разными для однозначности поиска источника прерывания. Регистры RG могут быть выполнены, например, в виде наборов микропереключателей, проводных перемычек либо в виде программно-доступных регистров обычного типа.

Сигнал разрешения BG проходит через логические элементы И тех внешних устройств, у которых триггер запроса ТТ установлен в единичное состояние (рис. 2.34, *а*) и в виде сигнала f поступает на входы соответствующих схем, включая их в процесс «конкурентной борьбы» на шине данных. Пока идет этот процесс, число конкурентов не должно расти, поэтому триггеры ТТ временно закрываются по входам (соответствующая аппаратура не показана для упрощения рисунка).

Логическая схема (рис. 2.34, *б*) имеет поразрядную структуру, напоминающую сумматор с последовательным переносом. Линии b_i шины данных реализуют функции Монтажное ИЛИ по отношению к лог. 1 (линии с общими эмиттерными резисторами). Фуикции, выполняемые одним разрядом логической схемы, описываются табл. 2.1.

В линию b_n старшего разряда выдают единицы все запросившие обслуживание внешние устройства, у которых в старшем разряде ре-

Таблица 2.1

a_n	b_n	$c_{вых}$	Примечание
0	0	1	Продолжить анализ следующего разряда
1	1	1	
0	1	0	Внешнее устройство выбывает из конкурса
1	0	—	Невозможная комбинация

гистра приоритета записана единица. При этом все внешние устройства, у которых $a_n=0$, не выдерживают конкуренции ($c_{вых}=0$) и исключаются из дальнейшего рассмотрения. Если нет ни одного внешнего устройства с $a_n=1$, то $b_n=0$ и конкурентная борьба переносится в следующий разряд. В линию b_{n-1} выдают единицы лишь те внешние устройства, которые прошли первый этап отбора и имеют единицу в разряде a_{n-1} и т. д.

Время формирования результирующего вектора не превышает $4\tau n$ (τ — задержка логического элемента) независимо от числа внешних устройств (увеличение паразитных емкостей линий b_i с увеличением числа внешних устройств для простоты не принимаем во внимание). В [24] показано, что время формирования результирующего кода можно уменьшить при разбиении разрядов на группы длиной $\sqrt{n}$ и введении схем группового переноса, как это делается в сумматорах.

Таким образом, время реакции на прерывание при схемном параллельном опросе существенно уменьшено по сравнению с временем реакции устройств, описанных в двух предыдущих примерах.

2.6. Канал прямого доступа в память

Канал прямого доступа по собственной инициативе или при запуске от микроЭВМ захватывает магистраль и обменивается информацией с памятью или с любыми адресуемыми элементами микроЭВМ. Обмен происходит либо в монопольном, либо в мультиплексном режиме. В монопольном режиме магистраль занимается на все время передачи массива, а в мультиплексном — лишь на время передачи очередного слова, затем освобождается, вновь занимается и т. д. Мультиплексный режим более «демократичен», поскольку он позволяет вычислительной машине продолжать работу по собственной программе и своевременно реагировать на внешние события. Монопольный режим временно приостанавливает работу микроЭВМ, что не всегда допустимо. Однако лишь монопольный режим обеспечивает необходимую скорость обмена при работе с «быстрыми» объектами. (О канале с «мгновенной» передачей информации, который не является ни мультиплексным, ни монопольным, будет сказано в конце этого параграфа.) Каналы прямого доступа обычно обеспечивают скорости передачи информации от 200 тыс. слов/с и выше.

На рис. 2.35 показан пример схемы контроллера прямого доступа. Блок регистров адреса предназначен для хранения конечного и те-

кущего адресов. Начальный и конечный адреса задаются из микроЭВМ на этапе «настройки» контроллера и определяют длину и расположение массива, подлежащего вводу или выводу. Регистр текущего адреса прибавляет единицу к старому содержимому после передачи каждого слова данных. Схема сравнения фиксирует момент окончания передачи массива и выдает сигнал в устройство управления. Регистр данных временно хранит передаваемое слово. Буфер данных вводится при необходимости согласования форматов данных, скоростей передачи и т. д. Регистр состояния отображает служебные признаки: готовность к работе, состояние схем контроля, тип выполняемого задания и др.

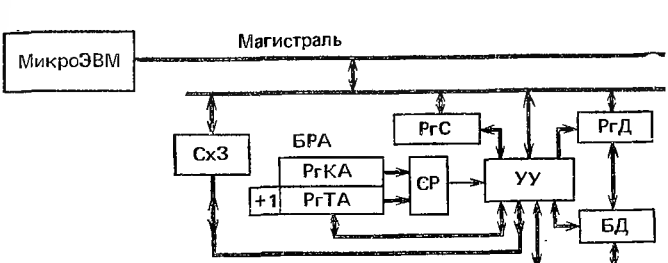


Рис. 2.35. Контроллер прямого доступа в память:

БРА — блок регистров адреса; РгТА — регистр текущего адреса; СР — схема сравнения; УУ — устройство управления; РгД — регистр данных; БД — буфер данных; РгС — регистр состояния; РгКА — регистр конечного адреса; СхЗ — схема захвата магистрали

В момент запуска контроллер является пассивным (см. классификацию пассивных контроллеров, § 2.3). Поэтому возможны три типа запуска:

1. Синхронный запуск: микроЭВМ «уверена», что канал готов к работе.

2. Асинхронный запуск: микроЭВМ периодически анализирует регистр состояния канала и дожидается его готовности к работе.

3. Запуск по прерыванию: канал сам сообщает процессору микроЭВМ о готовности.

По окончании работы канал устанавливает в единичное состояние один из разрядов своего регистра состояния, либо записывает определенный код в определенную ячейку памяти, либо посылает в микроЭВМ сигнал прерывания.

На рис. 2.36 показан канал прямого доступа, выполненный на основе микроЭВМ1; микроЭВМ2 является «центральной» машиной. (Для наглядности рисунка блоки памяти ОЗУ1 и ОЗУ2 обеих микроЭВМ «вынесены» на магистрали М1 и М2). Информация, предназначенная для передачи в ОЗУ2, поступает из канала связи К, обрабатывается процессором микроЭВМ1 и накапливается в ОЗУ1 (накопление информации условно отображено стрелкой 1). Затем эта информация с помощью процессора микроЭВМ1 пословно передается

в адаптер, который периодически захватывает магистраль М2 и пересылает очередное слово в ОЗУ2 (см. стрелку 2).

В процессе передачи массива информации из ОЗУ1 в ОЗУ2 (даже при отключении адаптера от магистрали М2 после передачи каждого слова) магистраль М2 не может в полной мере использоваться процессором микроЭВМ2 из-за конфликтов. В результате теряется производительность микроЭВМ2 и замедляется реакция на сигналы, поступающие от внешних объектов на входы контроллера, что в ряде случаев недопустимо.

Схема, показанная на рис. 2.37, лишена указанных недостатков [110]. Как и в предыдущей схеме, микроЭВМ1 готовит информацию,

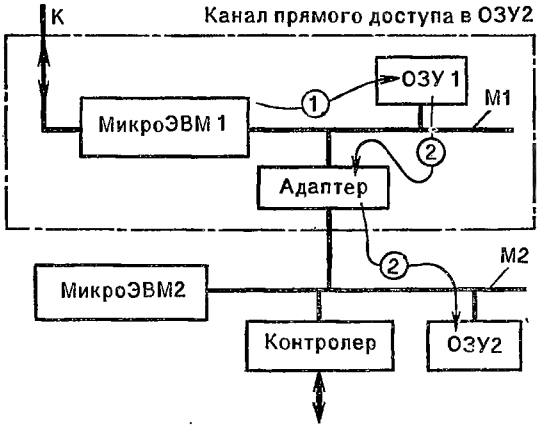


Рис. 2.36. Традиционная схема канала прямого доступа в память (ОЗУ2).

МикроЭВМ2 ощущает присутствие устройства-конкурента, сдерживающего темп обмена по магистрали М2

полученную из канала К, в блоке памяти ОЗУ1/2, подключенном к магистрали М1, для последующей передачи в «центральную» микроЭВМ2. После подготовки информации микроЭВМ1 «предлагает» микроЭВМ2 воспользоваться ею, т. е. микроЭВМ1 сообщает микроЭВМ2 о готовности информации выдачей сигнала прерывания F1.

Получив этот сигнал, микроЭВМ2 переводит переключатель SW (группу приемо-передающих логических элементов, управляемых от программно-доступного триггера) в положение Q2 и «мгновенно» получает в свое распоряжение весь подготовленный массив информации (вместе с ее физическим носителем). Иными словами, в свободную область адресного пространства микроЭВМ2 «вставляется» зона адресов, соответствующих ячейкам ОЗУ1/2 (подобно тому, как это делается с использованием «метода банков», см. § 1.5).

Обработав полученный массив информации, микроЭВМ2 посылает в микроЭВМ1 сигнал прерывания F2, по которому микроЭВМ1 возвращает переключатель SW в положение Q1, и затем готовит новый массив и т. д. Производительность микроЭВМ2 теряется лишь

в связи с обработкой однократных сигиалов прерывания (обработка прерывания может выполняться за время порядка 10 мкс). Если подключить второе ОЗУ (ОЗУ2/1), работающее в противофазе по отношению к ОЗУ1/2, как показано на рис. 2.37 штриховыми линиями, то можно обеспечить конвейерный режим работы, при котором обработка текущего массива процессором микроЭВМ2 совмещена по времени с подготовкой следующего массива процессором микроЭВМ1.

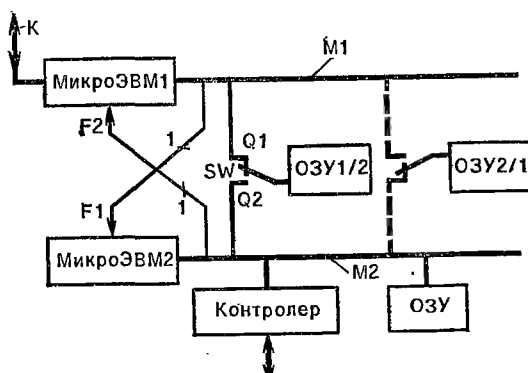


Рис. 2.37. Канал прямого доступа с «мгновенной» передачей информации.

За время порядка 10 мкс передается, например, 1М байт

Подведем основные итоги гл. 2.

1. Приступив к разработке контроллера для сопряжения микроЭВМ с объектом, определите его место согласно классификации, рис. 2.16. Попытайтесь использовать простейший тип канала — синхронный. Если он не подходит, перейдите к следующему — асинхронному, и т. д. Если ни один из каналов не подходит, то это, к сожалению, означает, что вы все еще до конца не уясиили задачу, которую предстоит решить.

2. Старайтесь «поручить» программе формирование управляющих импульсов во внутренних точках вашего контроллера, используя понятие программной доступности, как это сделано в рассмотренных примерах (см. § 2.4). Это позволит вам исключить из схемы элементы управления: одновибраторы, счетчики, линии задержки и т. п.

3. Используйте контроллер программируемого интерфейса (КПИ), если это возможно (а это почти наверняка так). Изменение интерфейса связи с объектом, подключение новых объектов и другие усовершеиствования повлекут за собой доработку программ и не затронут аппаратуру. Если же вы с самого начала отказались от КПИ, то, скорее всего, вам придется дорабатывать и то, и другое.

Глава 3

Последовательные каналы связи

3.1. Методы кодирования цифровой информации для ее передачи по последовательным каналам связи [9]

Передача информации между двумя достаточно удаленными устройствами требует представления ее в виде последовательного потока бит, характеристики которого зависят от особенностей конкретной системы. Физической основой такой системы является линия связи, которая обычно выполняется в виде витой пары проводов, коаксиального кабеля либо оптического световода. В зависимости от расстояния данные, передаваемые по линии, могут однократно или многократно подвергаться ретрансляции с целью восстановления амплитуды и временных характеристик (рис. 3.1).

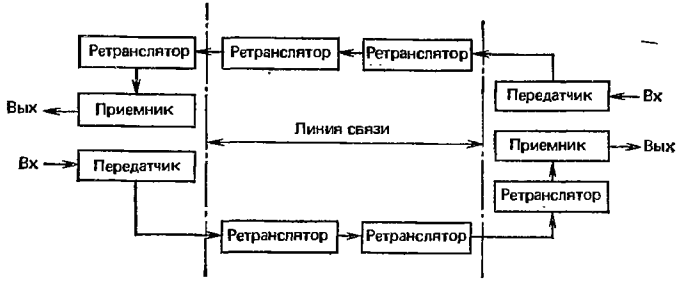


Рис. 3.1. Структура типового последовательного канала связи

Алгоритмы работы передатчика, ретранслятора и приемника определяются выбранным кодом, предназначенным для передачи по линии, или линейным кодом. Простейшим линейным кодом является униполярный код типа NRZ (non return to zero, рис. 3.2, *а*). В этом коде нули представлены отсутствием импульса (напряжение, близкое нулю), а единицы — наличием импульса (некоторое положительное напряжение). Этот код имеет четыре недостатка.

1. Средняя мощность, выделяемая на нагрузочном резисторе R, равна $A^2/2R$, где A — амплитуда импульса. Эта величина, как будет видно из дальнейшего, в два раза превышает мощность при «биполярном» кодировании.

2. Большинство линий связи сопрягаются с аппаратурой через реактивные элементы, такие, как трансформаторы. Поскольку униполярные сигналы всегда содержат постоянную составляющую и значительную долю низкочастотных компонентов в спектре при передаче длинной последовательности единиц, такое сопряжение затруднено или вовсе невозможно — реактивные элементы на достаточно низких частотах представляют собой либо «обрыв», либо «короткое замыкание».

3. Ретрансляторы и приемники способны надежно восстановить синхронизирующую временну́ю сетку только тогда, когда паузы между импульсами не слишком велики. Появление очередного импульса после незначительной паузы позволяет каждый раз корректировать «ход часов» ретранслятора или приемника. С увеличением паузы надежность «службы времени» этих устройств падает. Например, после передачи серии из 10 тыс. нулей приемник не сможет точно определить, находится ли последующая единица на позиции 9999, 10000

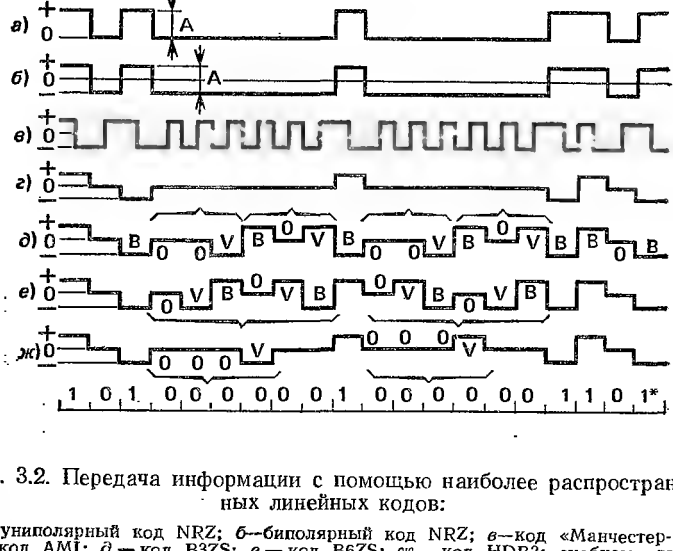


Рис. 3.2. Передача информации с помощью наиболее распространенных линейных кодов:

а — униполярный код NRZ; *б*—биполярный код NRZ; *в*—код «Манчестер-II»; *г* — код AMI; *д* — код B3ZS; *е* — код B6ZS; *ж* — код HDB3; скобками выделены «заготовки»

или 10001. Другими словами, при передаче достаточно большой последовательности нулей приемник (или ретранслятор) теряет синхронизацию с передатчиком (или ретранслятором).

4. Отсутствие возможности оперативной регистрации ошибок, таких, как пропадание или появление лишних импульсов из-за помех.

Биполярный сигнал NRZ (рис. 3.2, *б*) обладает лучшими энергетическими характеристиками. Единица представлена положительным уровнем напряжения, нуль — отрицательным. Средняя мощность равна $A^2/4R$, т. е. половине средней мощности униполярного сигнала, хотя перепад уровней тот же самый. Остальные три недостатка сохраняются. Для их ликвидации необходимо введение избыточности одним из двух способов:

1. Скорость передачи сигналов по линии выбирается большей, чем скорость передачи информации, без использования дополнительных электрических уровней сигналов.

2. Скорость передачи сигиалов по линии выбирается равной скорости передачи информации, однако вводятся дополнительные электрические уровни сигналов.

Примером кода с избыточностью, введенной согласно способу 1, является код «Манчестер-II». Форма биполярного сигнала при передаче кода «Манчестер-II» показана на рис. 3.2, *в*. Единица кодируется отрицательным перепадом сигнала в середине битового интервала, нуль — положительным перепадом. На границах битовых интервалов сигнал, если это небоходимо, меняет значеиие, «готовясь» к отображению очередного бита в середине следующего битового интервала

С помощью кода «Манчестер-II» решаются сразу все указанные проблемы. Поскольку число положительных и отрицательных импульсов на любом достаточно большом отрезке времени равно (отличается не более чем на один импульс, что не имеет значения), постоянная составляющая равна нулю. Подстройка часов приемника или ретранслятора производится при передаче каждого бита, т. е. снимается проблема рассинхронизации. Спектр сигнала содержит только две логические составляющие: F и 2F, где F — скорость передачи информационных бит. Наличие только двух (а не трех или более) электрических уровней напряжения позволяет надежно их распознавать (хорошая помехозащищеиность).

Критерием ошибки может являться «замораживаиие» сигнала на одном из уровней на время, превышающее время передачи одного информационного бита, поскольку независимо от передаваемого кода сигнал всегда «колеблется» и никогда не «замирает». Плата за эти чрезвычайно полезные качества — удвоение требуемой пропускной способности связной аппаратуры. Поэтому код «Маичестер-II» широко используется там, где частотные ограничения не являются определяющими, в частности во многих локальных сетях микроЭВМ.

Второй способ введения избыточности связан с добавлением дополнительных электрических уровней, в простейшем случае — третьего, «нулевого» уровия. На рис 3.2, *г* приведена форма сигнала с попеременной инверсией знака, так называемого AMI-сигнала (alternate mark inversion). Нули кодируются отсутствием импульсов, а единицы — попеременно положительными и отрицательными импульсами. Постояниая составляющая равна нулю, проблема передачи последовательности единиц отсутствует, обнаруживаются ошибки, иарушающие правильную последовательиость знакочередующихся сигналов.

Единственная оставшаяся проблема — потеря синхроиизации при передаче последовательиости нулей, как и в коде NRZ. Эта проблема решается очень просто: цепочки нулей передатчик заменяет определенными «заготовками» или «кусками» стандартных временных диаграмм (как заурядный конферансье заполняет паузы стандартными информационными посылками в зал). Коды AMI, в которых цепочка из N нулей заменяется определенной подстаиовкой, называются BNZS-кодами (bipolar with N zeroes substitution).

В коде B3ZS каждые три последовательио расположенных нуля подменяются либо комбинацией B0V, либо 00V. Символ B обозначает импульс, который отвечает правилам кодирования AMI. Символ V обозиачает импульс, который нарушает правила кодирования AMI (совпадает по полярности с предыдущим). Выбор одной из этих двух «заготовок» производится так, чтобы, во-первых, число импульсов B между двумя последовательно расположенными импульсами V бы-

ло нечетным и, во-вторых, чтобы полярность импульсов V чередовалась (рис. 3.2, *д*). В коде B6ZS каждые шесть последовательных нулей подменяются комбинацией 0VB0VB (рис. 3.2, *е*).

Коды BNZS получили широкое распространение в вычислительных сетях США и Канады [9]: линии T1 — 1,544 Мбит/с, T1C — 3,152 Мбит/с, LD-4 — 274,176 Мбит/с, T4 — 274,176 Мбит/с. В странах Западной Европы широко используется код HDB3 для работы на скоростях 2,048 и 8,448 Мбит/с. Этот код очень похож на BNZS, поскольку максимально допустимое число нулей, стоящих в цепочке, равно трем. Каждые четыре последовательно расположенных нуля подменяются комбинацией 000V либо B00V. Выбор той или иной комбинации производится так, чтобы, во-первых, число импульсов B между двумя последовательными импульсами V было нечетным и, во-вторых, чтобы полярность импульсов V чередовалась (рис. 3.2, *ж*).

Существуют также другие распространенные коды, такие, как CMI, PST, 4B3T и т. п. Все они являются разновидностями кодов AMI и созданы с целью минимизации требований к полосе пропускания каналов связи и увеличения обнаруживающей способности по отношению к ошибкам при передаче информации. Связная аппаратура для кодирования и декодирования информации относительно сложна. Шифратор-дешифратор кода HDB3 содержит 15 микросхем малой степени интеграции серии ТТЛ и потребляет мощность 0,8 Вт. Поэтому для этих целей созданы специальные микросхемы средней степени интеграции (CD22103, TCM2201, MJ1471 и др.), каждая из которых выполняет одновременно функции шифратора и дешифратора.

3.2. Примеры микроЭВМ с магистралями последовательного типа

МикроЭВМ, использующая код NRZ для связи процессора с контроллерами [26, 27]. На рис. 3.3 представлена схема микроЭВМ, содержащей центральный процессор (с памятью) и ряд удаленных контроллеров $К_i$, каждый из которых соединен с одним или несколькими периферийными устройствами (не показаны). Соединение составных частей микроЭВМ производится с помощью кабелей а — г. Каждый кабель содержит две витые пары проводов, возможно, в общей экранной оплетке. Первая пара проводов предназначена для передачи кода NRZ в ту или иную сторону, вторая — для передачи синхроимпульсов С от тактового генератора, расположенного в процессоре.

Положительный фронт сигнала С каждый раз подтверждает истинность очередного информационного бита, установленного на линии NRZ. Согласующие блоки содержат по два резистора R, замыкающих сигнальные и земляные провода линий NRZ и С. Согласующие блоки подключаются к физическим концам магистрали и служат для предотвращения отражений импульсов от концов кабеля. Сопротивление резисторов R выбирается равным волновому сопротивлению витой пары (110—130 Ом).

Формат информационной посылки при обмене информацией процессора с контроллером показан на рис. 3.4. Посылка содержит стартовый бит, код операции КОП или «команду» и данные D0—D7. Каждый контроллер имеет собственный 8-разрядный адрес от 1 до

255_{10} (код 0 не используется). В магистрали определены следующие четыре типа обменов (команд).

1. Выбор контроллера (КОП=001). В разрядах D0—D7 процессор выставляет адрес того контроллера, с которым он желает связаться.

2. Запись данных в контроллер, выбранный командой 1 (КОП=111). Данные D0—D7 поступают из процессора в контроллер. Эта

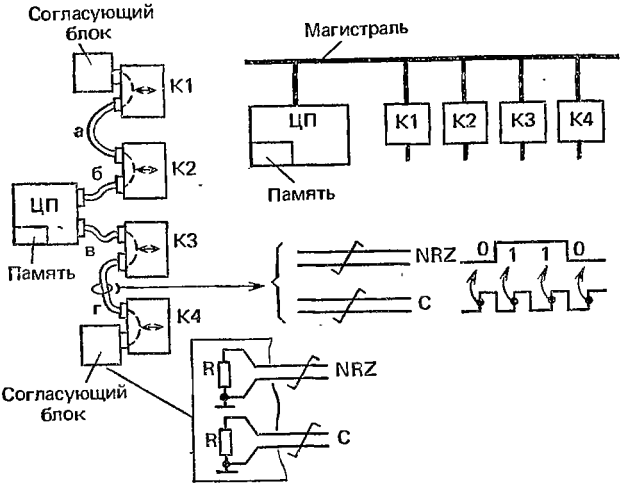


Рис. 3.3. «Физическая» и структурная схемы микроЭВМ с магистралью последовательного типа

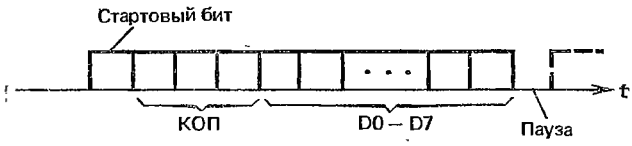


Рис. 3.4. Формат информационной посылки по магистрали микроЭВМ

команда может иметь ряд модификаций, отличающихся кодами операций. В зависимости от кода операции данные записываются либо в информационный регистр контроллера, либо в его регистр состояния, либо в один из служебных регистров.

3. Чтение данных из контроллера, выбранного командой 1 (КОП=110). Стартовый и три командных бита поступают в магистраль из процессора; затем он переключается на прием, а данные D0—D7 выдаются выбранным ранее (командой с КОП=001) контроллером. Эта команда также может иметь ряд модификаций.

4. «Приглашение к прерыванию» (КОП=000). Стартовый и три командных бита поступают в магистраль из процессора; затем он переключается на прием, а данные D0—D7, считываемые процессором с магистрали, представляют собой адрес наиболее приоритетного контроллера из числа требующих обслуживания. Этот адрес используется процессором как вектор прерывания, т. е. служит для указания начального адреса программы обслуживания данного контроллера.

Первые три команды реализуются в аппаратуре достаточно просто—с использованием сдвиговых регистров для синхронной последовательной выдачи (со стороны передатчика) и приема (со стороны приемника) информации согласно рис. 3.4. В исходном состоянии и во время пауз напряжение на линии NRZ близко к нулю. Появление первого единичного сигнала расценивается всеми устройствами как начало посылки (стартовый бит). Все устройства «знают», что после стартового бита следуют три командных бита, а затем восемь позиций для передачи информации в ту или иную сторону, в зависимости от значения кода операции КОП. После передачи по магистрали последнего бита D7 следует пауза, как минимум, в один период сигнала С. Следующий после паузы единичный сигнал на линии NRZ вновь рассматривается всеми устройствами как стартовый бит и т. д.

Рассмотрим теперь последний тип обмена — «приглашение к прерыванию». Из-за отсутствия специальной линии запроса на прерывание слежение за состоянием контроллеров производится самим процессором. Процессор периодически, например через каждые 0,5 мс, выдает в магистраль команду приглашения к прерыванию (КОП=000). Если претендентов на обслуживание нет, то в считанных процессором разрядах D0—D7 окажутся нули. Как отмечалось, нулевой код не используется, поэтому процессор «узнаёт», что ни один из контроллеров не требует обслуживания. Если есть единственный претендент на обслуживание, то процессор принимает его адрес на позициях D0—D7. Если претендентов на обслуживание несколько, то они сами, без участия процессора, решают, кто из них «главнее». (Вспомните работу схемы, приведенной на рис. 2.34. Здесь принцип тот же, но теперь он воплощен в «последовательном» варианте.) «Побеждает» контроллер с бо́льшим собственным адресом.

Предположим, что в конкурентную борьбу по команде с КОП=000 включились три контроллера со следующими собственными адресами:

```
1 0 1 1 0 0 0 1  — контроллер магнитофона
1 0 1 0 1 1 0 1  — контроллер печатающего устройства
0 1 1 1 0 1 1 1  — контроллер считывателя с перфокарт
-----------------
1 0 1 1 0 0 0 1  — результирующий код в магистрали
D0 ... D3  ...  D7   (производится поразрядное суммирование по ИЛИ).
```

В разряд D0 выдают единицы первые два контроллера (линия NRZ в данном примере реализует функцию ИЛИ по отношению к единичным сигналам). Третий контроллер, «увидев» в магистрали единицу и зная, что старший разряд его собственного адреса равен нулю, переходит в пассивное состояние, выбывает из конкурса и ждет новой команды приглашения к прерыванию. В последующие моменты, соответствующие позициям D1 и D2, в линию NRZ выдаются нули,

а затем единицы из обоих оставшихся контроллеров. В момент, соответствующий позиции D3, в линию выдается единица из контроллера магнитофона. При этом второй контроллер, увидев в линии единицу и сравнив ее с нулем в разряде 3 собственного адреса, выбывает из конкурса. Оставшиеся разряды выдаются в линию из контроллера магнитофона, так как все конкуренты выбыли из борьбы (см. зачеркнутые строки в перечне адресов контроллеров).

Таким образом, в процессор поступает адрес первого контроллера. Процессор, определив, что полученный адрес не равен нулю, выполняет стандартную процедуру записи в свою память параметров

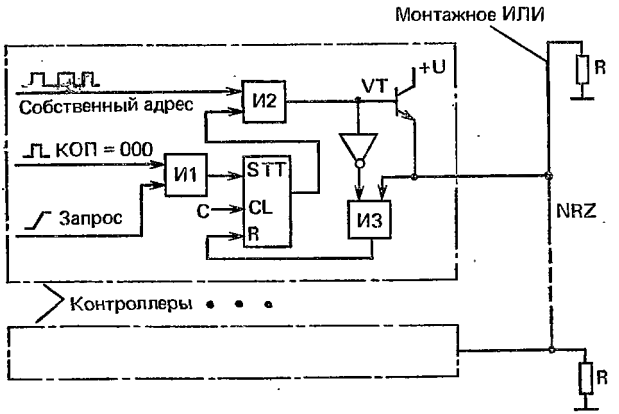


Рис. 3.5. Фрагмент приоритетной схемы, расположенной в контроллере

возврата и переходит к соответствующей программе обслуживания, после выполнения которой он возвращается к прерванной задаче.

В процессе выполнения программы обслуживания выбранного контроллера механизм периодической выдачи команд приглашения к прерыванию может либо действовать, либо нет — все зависит от программиста. В первом случае прерывающая программа может быть, в свою очередь, прервана другими запросами, во втором — реакция на запросы затормаживается до тех пор, пока не будет выполнена первая прерывающая программа.

Схемная реализация приоритетного взаимодействия контроллеров при выполнении команды приглашения к прерыванию приведена на рис. 3.5. При обнаружении в линии NRZ кода операции КОП=000 и при наличии собственного запроса срабатывает логический элемент И1, в результате триггер ТТ устанавливается в 1, разрешая выдачу в линию NRZ последовательного кода собственного адреса через логический элемент И2 и эмиттерный повторитель (передатчик) на транзисторе VT.

Если текущий разряд собственного адреса равен нулю, а в линию NRZ кто-либо из конкурентов выдал единицу, то срабатывает логический элемент И3 и триггер устанавливается в 0 по входу R, препятствуя дальнейшей выдаче разрядов адреса в линию. Таким об-

разом, контроллер выбывает из конкурса. Если же конкуренты отсутствуют или имеют более низкие приоритеты, то адрес выдается полностью, после чего триггер сбрасывается по внутренней цепи со счетчика тактов (эта цепь для простоты не показана).

Число проводов в магистрали можно сократить до двух (один сигнальный и один «земляной»), исключив канал передачи синхросигнала С. В этом случае в каждом контроллере должны быть собственные «часы», которые запускаются фронтом стартового импульса на линии NRZ (рис. 3.6).

Рис. 3.6. Временная диаграмма передачи посылки в отсутствие линии передачи синхросигналов

Для надежного приема информационной посылки из процессора собственные часы контроллера должны формировать метки времени, попадающие в середины битовых интервалов. С течением времени, начиная от момента запуска часов, происходит накопление ошибки. Из рис. 3.6 видно, что максимальная ошибка 12,5Δ не должна достигать половины длительности Т импульса: $\Delta < 0{,}5T/12{,}5 = 0{,}04T$ или, другими словами, рассогласование хода часов передатчика и приемника не должно превышать 4 % даже без учета разброса времени запуска.

Для обеспечения нужной точности частота генератора часов выбирается обычно в 16, 32 или 64 раза большей, чем частота следования информационных бит в посылке. С увеличением скорости передачи и длины посылки, а также расстояния между устройствами надежность передачи снижается. Однако из-за простоты такой способ передачи информации получил широкое распространение — стандарты RS—232 (США), ИРПС (СССР). Скорости передачи невелики: 110, 150, 300, 600, 1200, 2400, 4800, 9600, 19 200 бит/с, расстояние между устройствами зависит от скорости и типа линии и обычно не превышает 0,5—1 км. Информация в линии представляется либо разнополярными уровнями напряжений, либо наличием или отсутствием тока (20- или 40-миллиамперные «токовые петли»). Подобные каналы связи часто называют «телеграфными каналами» из-за сходства структуры информационных посылок (но не электрофизических характеристик).

В [26] отмечается, что кабели а—г (см. рис. 3.3) могут быть вообще исключены из микроЭВМ. Их может заменить «эфир», если воспользоваться световым, ультразвуковым либо радиоканалом. При этом, конечно, целостность микроЭВМ сохраняется и, в частности, остается в силе механизм приоритетного взаимодействия контроллеров.

МикроЭВМ с малым временем реакции на прерывание [28]. Время реакции на прерывание для многих систем является существенным показателем качества их работы. В предыдущем примере (см. рис. 3.3—3.5) даже при отсутствии конкурентов контроллер может опоздать к моменту приглашения к прерыванию. В этом случае ему придется ждать следующего такого приглашения, например 0,5 мс, что не всегда допустимо. Увеличение частоты следования команд приглашения к прерыванию не решает проблемы, так как при этом все бóльшая часть машинного времени тратится впустую. Использование отдельной двухпроводной линии связи контроллера с процессором для передачи запроса на прерывание нежелательно, особенно при больших расстояниях.

Рассмотрим простую схему связи, позволяющую по одной и той же двухпроводной линии передавать как запрос на прерывание, так

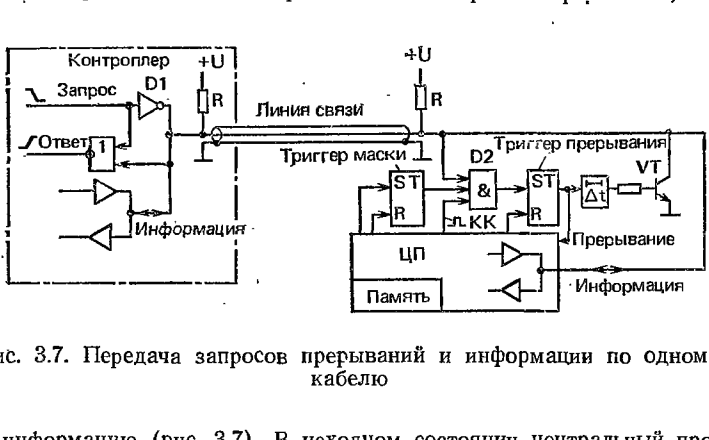


Рис. 3.7. Передача запросов прерываний и информации по одному кабелю

и информацию (рис. 3.7). В исходном состоянии центральный процессор работает по программе, не связанной с обменом. Триггер маски установлен в единичное, т. е. в разрешающее состояние. Триггер прерывания сброшен. Напряжение низкого уровня в линии связи поддерживается только благодаря насыщенному выходному транзистору инвертора D1, все остальные передающие элементы (построенные по схеме с открытым коллектором) выключены. Сигнал ответа отсутствует, т. е. равен нулю. Сигнал запроса отсутствует, т. е. равен единице, поскольку контроллер не требует обслуживания. После выполнения каждой команды из устройства управления процессора на вход элемента D2 поступает импульс КК (конец команды), который опрашивает состояние линии связи. Поскольку в исходном состоянии запрос на прерывание отсутствует (в линии — напряжение низкого уровня), сигнал установки в 1 триггера прерывания не вырабатывается.

При появлении запроса на обслуживание инвертор D1 выключается и в линии связи благодаря резисторам R устанавливается напряжение высокого уровня (лог. 1). После выполнения текущей команды формируется импульс КК, который, пройдя через элемент D2, устанавливает триггер прерывания в 1, заставляя процессор начать переход к прерывающей программе по фиксированному вектору (вектор задан в самóм процессоре).

В то же время единичный сигнал с выхода триггера прерывания поступает в элемент задержки и затем включает транзистор VT, в результате в линии формируется напряжение низкого уровня (лог. 0). На выходе логического элемента ИЛИ — НЕ вырабатывается ответный сигнал лог. 1, который сообщает контроллеру о том, что процессор принял запрос и готовится к обслуживанию контроллера. Далее процессор выключает триггер маски и триггер прерывания (оба триггера выполнены программно-доступными), транзистор VT закрывается, в линии устанавливается напряжение высокого уровня, ответный сигнал с выхода элемента ИЛИ — НЕ снимается, контроллер готов принять информацию по линии связи из процессора. Процессор приступает к обмену, используя, например, «телеграфный» код и команды, аналогичные командам 2 и 3, рассмотренным в предыдущем примере (см. рис. 3.3—3.5).

По окончании обмена запрос снимается, т. е. на входе инвертора D1 формируется лог. 1. Это происходит, например, при записи процессором лог. 1 в определенный разряд регистра состояния контроллера (не показан) при выполнении последней команды программы обмена. При этом в линии связи устанавливается напряжение низкого уровня за счет срабатывания инвертора D1. Далее, если это необходимо, процессор устанавливает триггер маски в исходное (разрешающее) состояние по входу S. Таким образом, микроЭВМ переходит в исходное состояние, при котором процессор возвращается к фоновой задаче, а линия связи подготовлена к приему очередного сигнала прерывания.

МикроЭВМ не может использовать линию связи по собственной инициативе — сеанс возможен только по инициативе контроллера. Еще один недостаток: неясно, как наращивать число контроллеров. Возможный путь решения этих проблем описан в § 5.3.

3.3. Каналы, использующие код «Манчестер-II»

Канал связи (рис. 3.8) включает шифратор, дешифратор и двухпроводную магистраль. Сигнал в коде «Манчестер-II» может быть получен суммированием по модулю два сигналов NRZ и C. Другими словами, сигнал, представленный в коде «Манчестер-II», принимает единичные значения в тех интервалах времени, в которых сигналы NRZ и C имеют противоположные логические значения (01 или 10). Вследствие этого схема шифратора кода «Манчестер-II» чрезвычайно проста (рис. 3.9). Схема подавления помех предназначена для «очистки» результирующего сигнала «Манчестер-II» от кратковременных импульсов, которые могут возникнуть из-за неидеального совпадения отрицательного фронта сигнала C с отрицательным или положительным фронтом сигнала NRZ.

Дешифратор кода «Манчестер-II» представляет собой несколько более сложную схему [30], содержащую формирователь импульсов (ФИ), счетный триггер (СТ) и D-триггер (рис. 3.10, *а*). Как следует из временной диаграммы, приведенной на рис. 3.10, *б*, отрицательные импульсы a на выходе формирователя импульсов возникают всякий раз, когда сигнал «Манчестер-II» меняет свое значение (0→1 или 1→0). Сигнал b восполняет «недостающие» импульсы a. (Временную диаграмму сигнала b можно нарисовать, используя только диаграмму сигнала a, следовательно, сигнал b можно получить из сигнала a с помощью логической схемы [30].)

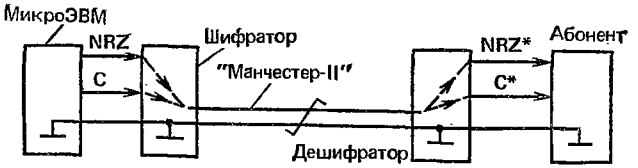


Рис. 3.8. Канал, использующий код «Манчестер-II»

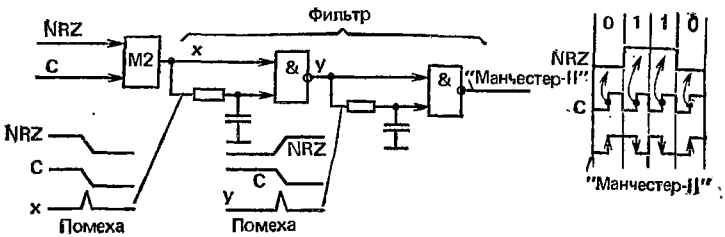


Рис. 3.9. Шифратор кода «Манчестер-II»

Так как импульс b поступает на установочный вход S счетного триггера СТ, то в момент t_0 этот триггер обязательно перейдет в единичное состояние и в дальнейшем сигнал C*, снимаемый с его инверсного плеча, будет в точности повторять сигнал C, выдаваемый микроЭВМ (разумеется, с некоторой задержкой).

Начиная с момента t_1, т. е. по прошествии одного периода тактовых импульсов от момента t_0, код NRZ*, снимаемый с выхода триггера ТТ, полностью совпадает с кодом NRZ, поступающим из микроЭВМ на шифратор (с точностью до задержки передачи). Таким образом, для того чтобы заставить приемник войти в синхронизм с передатчиком, достаточно на линии NRZ создать переход сигнала из 0 в 1. Последующая цепочка бит любой длины, передаваемая по линии NRZ, будет в точности повторена на линии NRZ* приемника. Это же относится и к синхросигналам: сигнал C* в точности повторяет исходный сигнал C.

Остановимся теперь более подробно на преимуществах кода «Манчестер-II» перед кодом NRZ.

1. Синхросигналы и информация передаются по одному каналу, в то время как при использовании кода NRZ нужны два канала (сравнение с одноканальным или «телеграфным» вариантом будет проведено отдельно).

2. Диапазон логических частот NRZ начинается от нуля и не превышает половины тактовой частоты (рис. 3.11). Сигнал «Манчестер-II» содержит только две логические составляющие: $f_c/2$ и f_c. Постоянная составляющая при использовании биполярных сигналов равна нулю. Из этого следует, что приемник кода «Манчестер-II» может быть узкополосным и поэтому более помехоустойчивым, чем приемник кода NRZ. Кроме того, легко реализуется трансформаторная связь отдельных устройств с общей двухпроводной магистралью, как это сделано для бортовой микроЭВМ, описанной в [31] (рис. 3.12).

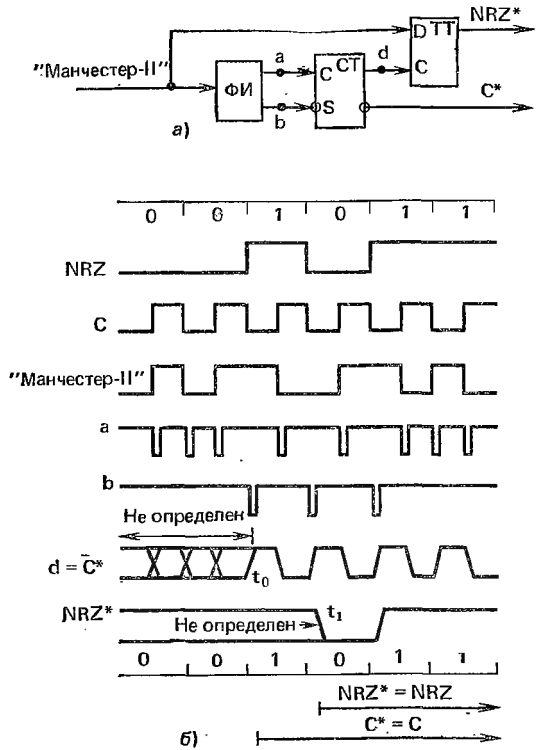


Рис. 3.10. Дешифрация кода «Манчестер-II»:
а — схема дешифратора; *б* — временные диаграммы работы дешифратора

Согласно стандарту MIL-STD-1553B (США) число устройств, подключаемых к такой магистрали, достигает 31 при общей длине кабелей не более 91 м. Амплитуда сигналов в магистрали 6—9 В. При отказе любой ветви (обрыв или короткое замыкание в ответвлении L) система в целом остается работоспособной. Скорость передачи информации по магистрали — 1 Мбит/с. Подобная система связи по уже описанным причинам непримениима для передачи кода NRZ.

3. Критерий ошибки передачи — наличие постоянного уровня сигнала в течение времени, превышающего один период тактовой частоты (в коде NRZ подобного критерия не существует). Однако в некоторых случаях [31] стартовые импульсы искусственно «затягиваются» на 1,5 периода, чтобы абоненты расценили эту ситуацию как начало сообщения. При этом, конечно, критерий ошибки пересматривается.

Основное преимущество кода «Манчестер-II» перед «телеграф-

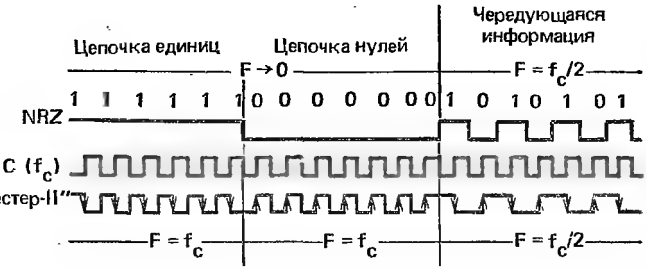


Рис. 3.11. Сравнение частотных характеристик сигналов NRZ и «Манчестер-II»

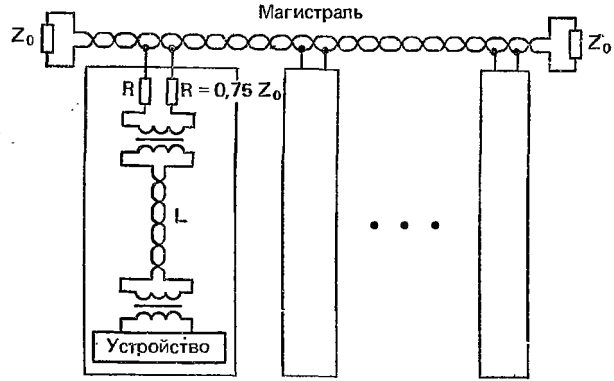


Рис. 3.12. Двухпроводная магистраль, ориентированная на работу с кодом «Манчестер-II»

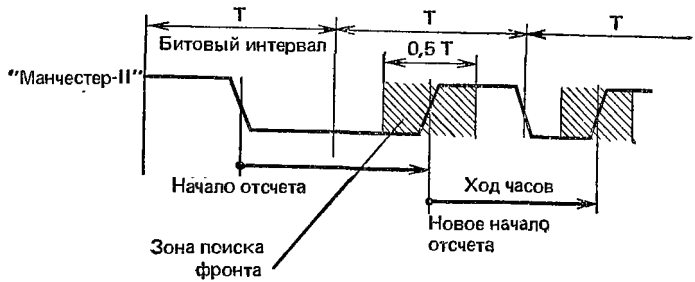


Рис. 3.13. Временна́я диаграмма передачи кода «Манчестер-II», обеспечивающая коррекцию хода часов приемника при передаче каждого бита

ным» кодом — побитовая синхронизация. Находясь в середине битового интервала и обнаружив фронт сигнала «Манчестер-II» (он обязательно должен быть: положительный либо отрицательный), можно корректировать ход своих часов и ожидать следующий информационный фронт через время, равное одному периоду (в некотором доверительном интервале). Зафиксировав следующий фронт, перенесем начало отсчета времени вправо на один такт и т. д. (рис. 3.13). Таким образом, начало отсчета каждый раз переносится на одну битовую позицию и ошибка не накапливается. Рассогласование хода часов передатчика и приемника может достигать 25 %, а не 4 %, как в «телеграфном» канале, и не зависит от длины посылки. За стартовым битом могут следовать, например, 10 000 информационных битов — адреса, данные, управляющие слова и т. д.

Недостатки канала для передачи кода «Манчестер-II» по сравнению с каналом для передачи кода NRZ: наличие схем шифрации и дешифрации и удвоенная по сравнению с необходимой пропускная способность.

Использование кода «Манчестер-II» для передачи информации по волоконно-оптической линии связи обеспечивает возможность работы светоизлучающего элемента с двукратной перегрузкой по выделяемой в импульсе мощности, так как в среднем 50 % времени этот элемент находится в выключенном состоянии. (Сигнал, как отмечалось, никогда не «останавливается» в том или ином положении более чем на один такт, а скважность в среднем равна двум.)

3.4. Каналы для одновременной двунаправленной передачи информации по одному кабелю

В дисплее VT100 (фирма DEC, США) использована схема, позволяющая передавать информацию одновременно в двух направлениях от двух независимых и не синхронизированных между собой источников (рис. 3.14). Схема содержит два одинаковых приемопередатчика, объединенных двухпроводной линией связи L. В табл. 3.1 показаны четыре возможные состояния устройства. Сигналы A, B, C и D представлены в табл. 3.1 логическими значениями; остальные сигналы представлены в виде приблизительных значений напряжений (в вольтах).

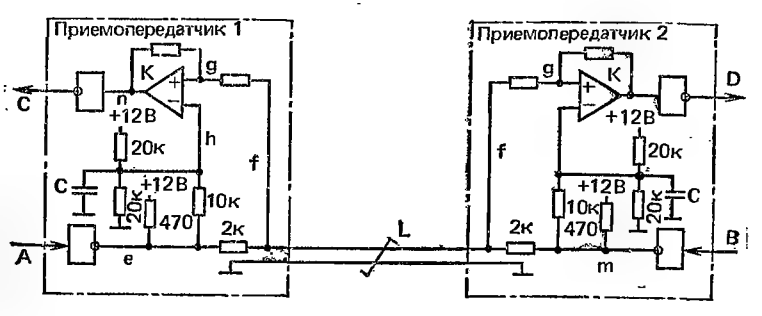


Рис. 3.14. Канал связи процессора с блоком клавиатуры

Таблица 3.1

A	B	e	m	g=f	h	g—h	n	C	D
0	0	+12	+12	+12	+9	+3	+5	0	0
0	1	+12	0	+ 6	+9	—3	0	1	0
1	0	0	+12	+ 6	+3	+3	+5	0	1
1	1	0	0	0	+3	—3	0	1	1

Для упрощения рассуждений будем считать, что в каждом приемопередатчике резистор, включенный между выходом и входом «+» компаратора К, а также кондеисатор исключены из схемы. Эти элементы влияют на динамические характеристики канала связи: резистор создает гистерезисную обратную связь, улучшая крутизну фронтов и повышая помехоустойчивость, а конденсатор выравниваст время нарастания (и спада) сигналов на входах «+» и «—» компаратора, компенсируя таким образом емкость линии связи и устраняя ложные «перекосы» сигналов на фронтах. Сделав такие допущения, можно считать, что схема «статична», и рассчитать напряжения во всех ее внутреиних точках, пользуясь законом Ома. Рассмотрим, например, первую строку табл. 3.1. При подаче сигналов лог. 0 на входы А и В канала связи на выходах е и m входных инверторов с «открытым коллектором» формируются напряжения, близкие +12 В. Поступая через резисторы сопротивлением 10 кОм в средние точки соответствующих резисторных делителей, они, в свою очередь, повышают напряжения в этих точках примерно до +9 В. Напряжение на сигиальном проводе f линии связи в этом случае равно примерно +12 В. Разность напряжений (g—h), подаваемых на входы «+» и «—» компаратора приемопередатчика 1, положительна и составляет примерно 3 В, поэтому на его выходе в точке п формируется сигнал амплитудой около 5 В, а на выход С инвертора выдается лог. 0. В силу симметрии канала и равенства логических значений входных сигиалов иа выходе D приемопередатчика 2 также формируется сигнал лог. 0.

Продолжая аналогичные рассуждения, можно убедиться в истинности информации, приведенной в табл. 3.1. Из этой таблицы следует, что D=A и C=B при всех возможных ситуациях. Другими словами, два потока информации идут навстречу друг другу без взаимодействия, как если бы они шли по двум независимым однонаправленным каналам. Выходные инверторы можно исключить, при этом канал будет инвертировать передаваемую информацию.

Та же задача встречной одновременной передачи информации по одной паре проводов может быть решена с использованием другой схемы (рис. 3.15) с лучшими динамическими характеристиками и с гальванической развязкой между линией связи и устройствами — участниками обмена.

Рассмотрим сначала работу составных частей схемы. Усилители D3 и D4 представляют собой буферные логические элементы ТТЛ с открытым коллектором и предназначены для управления светодиодами оптронов V2, V3 и V6, V7. При A=0 (на входе напряжение низ-

кого уровня) выходной транзистор элемента D3 насыщен, светодиод оптрона V3 излучает свет, светодиод оптрона V2 выключен. При A=1 выходной транзистор элемента D3 закрыт, светодиод оптрона V3 выключен, светодиод оптрона V2 излучает свет. Управление светодиодами оптронов V6 и V7 производится аналогично.

Передающие оптроны V2, V3, V6 и V7 предназначены для передачи информации в линию L. Если потенциал коллектора выходного транзистора оптрона V2 (V3, V6, V7) более положителен, чем потенциал эмиттера, и на этот транзистор поступает световой поток от соответствующего светодиода, то транзистор переходит в состояние насыщения, при котором коллектор «замыкается» с эмиттером. Во

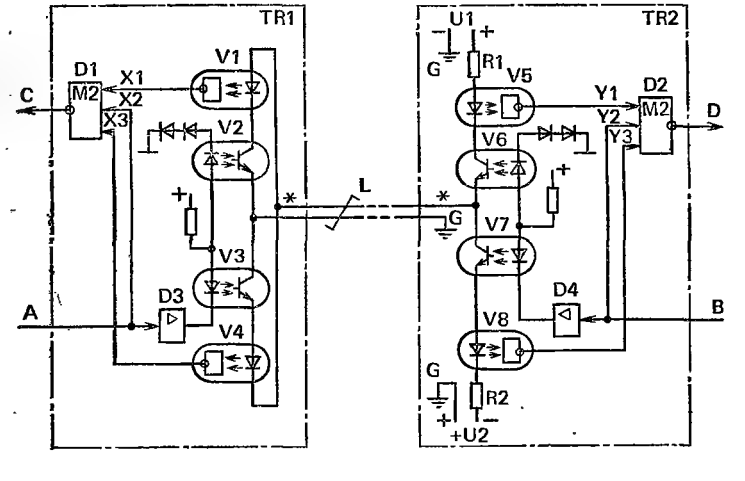


Рис. 3.15. Канал для одновременной двунаправленной передачи информации по одному кабелю с гальванической развязкой приемопередатчиков

всех остальных случаях (при отсутствии светового потока или (и) при обратном включении транзистора) транзистор выключен, т. е. его коллектор «оборван» внутри оптрона.

Приемные оптроны V1, V4, V5, V8 предназначены для приема информации из линии L. Светодиод приемного оптрона V1 (V4, V5, V8) включается в том случае, когда потенциал его анода более положителен, чем потенциал катода, и, кроме того, через него протекает достаточно большой ток (например, 10 мА). Если ток через светодиод при прямом смещении близок нулю (при размыкании последовательно включенного с ним транзистора передающего оптрона) или если на светодиод подано обратное смещение, то свет не излучается. Если светодиод оптрона V1 (V4, V5, V8) излучает свет, то на его выходе X1 (X3, Y1, Y3) формируется сигнал лог. 0; в противном случае — сигнал лог. 1.

Элементы D1 и D2 выполняют суммирование по модулю два

входных сигналов с последующим инвертированием полученной суммы. Другими словами, сигнал лог. 1 вырабатывается на выходе элемента D1 (D2) только тогда, когда число единичных сигналов на его входах четно.

Линия связи L гальванически развязана с входами и выходами схемы. Питание линии производится от двух источников напряжения: U1 (с заземленным отрицательным полюсом) и U2 (с заземленным положительным полюсом). Точка заземления G линии L гальванически развязана со «схемными землями» приемопередающих блоков TR1 и TR2. Источники напряжения U1 и U2 выполняются гальванически развязанными со «штатными» источниками питания блоков TR1 и TR2. Это позволяет повысить помехоустойчивость канала связи. Линию L предпочтительно выполнять витой парой проводов.

Блоки TR1 и TR2 в функциональном отношении равноправны. Различие между ними лишь в том, что источники питания находятся в одном из них, в данном примере — в блоке TR2.

Работа схемы поясняется табл. 3.2, в которой показаны четыре возможных ее состояния при различных комбинациях входных сигналов.

Таблица 3.2

A	B	X1	X2	X3	C	Y1	Y2	Y3	D
0	0	1	0	0	0	1	0	0	0
0	1	1	0	1	1	1	1	1	0
1	0	1	1	1	0	1	0	1	1
1	1	0	1	1	1	0	1	1	1

Из табл. 3.2 следует, что выходные сигналы совпадают с входными, поступающими с противоположной стороны, т. е. сигнал на выходе C в любой ситуации повторяет сигнал на входе B, а сигнал на выходе D совпадает с сигналом, поданным на вход A.

Первая строка табл. 3.2 соответствует подаче на входы A и B сигналов лог. 0. В этом случае, как было показано, оптроны V2 и V6 выключены, а оптроны V3 и V7 включены. Это создает благоприятные условия для протекания тока по цепи источника U2: G—V3—V4—V7—V8—R2, поэтому срабатывают оптроны V4 и V8 и на их выходах X3 и Y3 формируются сигналы лог. 0. Оптроны V1 и V5 выключены, поэтому на их выходах X1 и Y1 формируются сигналы лог. 1. Таким образом, число единиц, поданных на входы элементов D1 и D2, в данном случае нечетно (у каждого элемента по одной единице), следовательно, на выходах C и D формируются сигналы лог. 0.

Вторая и третья строки табл. 3.2 соответствуют подаче разноименных сигналов на входы устройства. В обоих случаях ток в линии L отсутствует, хотя светодиоды оптронов V3 и V6 или V2 и V7 излучают свет (вспомните об условиях протекания тока через выходной транзистор передающего оптрона, рассмотренных ранее).

Логика работы схемы в этих двух режимах такова. Если блок TR1 (TR2) «увидел», что тока в линии L нет, то он узнаёт, что с про-

тивоположной стороны передается сигнал, логически противоположный его собственному входному сигналу A (B). Поэтому сигнал на выходе C (D) может быть получен инвертированием «своего» сигнала, что и делается с помощью элемента D1 (D2), который при X1= =X3=1 (Y1=Y3=1) реализует функцию $C=\overline{A}$ ($D=\overline{B}$).

Четвертая строка табл. 3.2 соответствует подаче на входы схемы сигналов лог. 1. В этом случае срабатывают оптроны V2 и V6 и создаются условия для протекания тока по цепи источника U1 : R1—V5—V6—V1—V2—G. На выходах оптронов V1 и V5 формируются сигналы X1=0 и Y1=0, при этом в отсутствие тока через оптроны V4 и V8 на их выходах формируются единичные сигналы X3=Y3=1. Поэтому элементы D1 и D2, как и в первом режиме, передают сигнал A на выход C и сигнал B на выход D без инвертирования.

Переходные процессы в устройстве могут сопровождаться появлением кратковременных ложных импульсов на выходах C и D. Эти импульсы могут быть подавлены, например, с помощью интегрирующих RC-цепей, подключаемых к этим выходам. Альтернативным решением может являться многократный программный опрос выходов с выявлением неустойчивых логических значений сигналов, как это обычно делается при программной компенсации «дребезга» контактов механических переключателей. В ряде случаев эти ложные импульсы могут и не нарушать работоспособности системы. Например, два встречно включенных (через данную схему) телетайпа не реагирует на кратковременные ложные импульсы в силу своей механической инерционности. Схема с некоторыми отличиями от рассмотренной приведена в [111].

3.5. Канал для одновременной передачи напряжения питания, синхроимпульсов и информации по одному кабелю (по материалам [32])

На рис. 3.16 приведена схема канала связи для передачи показаний температурного датчика на цифровой индикатор. Питание части схемы, непосредственно связанной с датчиком, осуществляется от источника напряжением +24 В через резистор сопротивлением 620 Ом, сигнальный провод линии связи L и диод. По этому же проводу через транзистор VT1 проходят кратковременные «отрицательные» импульсы в кратковременные периоды открывания этого транзистора сигналом CL. Скважность импульсов CL выбирается большей 10. Эти импульсы не проходят через диод и, следовательно, не нарушают режима питания схемы, связанной с датчиком. В результате включения стабилизатора в точке F поддерживается стабилизированное напряжение +5 В, необходимое для правильной работы датчика, цифро-аналогового преобразователя и компаратора K1.

В исходном состоянии в момент t_0 в вычитающем счетчике установлен код 1111111_2, на выходе цифро-аналогового преобразователя устанавливается напряжение, близкое +5 В, компаратор K1 формирует на выходе нулевой сигнал, так как потенциал на его входе «—» более положителен, чем на входе «+». Транзистор VT2 закрыт. В дальнейшем, по мере уменьшения содержимого вычитающего счетчика, потенциал входа «—» компаратора K1 снижается и, наконец, в момент t_1 становится более отрицательным, чем потенциал входа «+». Транзистор VT2 включается и потребление тока по линии связи L резко увеличивается.

Этот факт регистрируется компаратором К2, на выходе которого формируется сигнал В=0, разрешающий работу суммирующего счетчика, первоначально установленного в 0. Суммирующий счетчик начинает накопление единиц (синхроимпульсов CL) до момента t_2, при котором в вычитающем счетчике вновь устанавливается код 1111111 и т. д. В момент t_2 содержимое суммирующего счетчика переписывается в выходной регистр и отображается на индикаторе, а этот счетчик через время Δt устанавливается в 0.

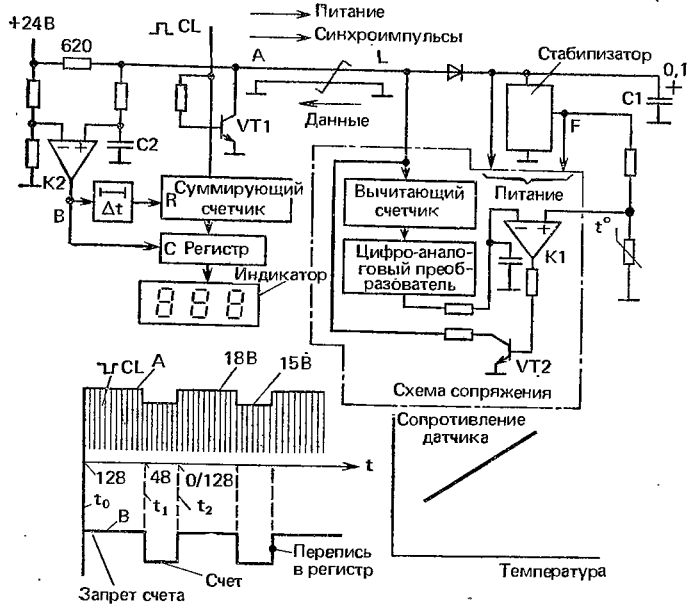


Рис. 3.16. Схема связи регистрирующего прибора с удаленным датчиком

Далее цикл работы повторяется: схема, связанная с датчиком, в некоторый момент в соответствии с показаниями датчика скачком увеличивает потребление тока, после чего включается суммирующий счетчик, а затем новые данные выдаются на индикатор. Конденсатор С2 предназначен для фильтрации синхроимпульсов. Чем выше температура, тем раньше включается транзистор VT2 и, следовательно, тем больше импульсов регистрируется суммирующим счетчиком. Длина линии связи — 75 м.

Приведенную схему не следует рассматривать узко, только как «схему связи с температурным датчиком». Она представляет собой пример творческого подхода к решению задачи миимизации числа связей между удаленными друг от друга устройствами.

3.6. Асинхронные двунаправленные приемопередатчики

При распространении сигнала вдоль линии связи его форма искажается: «заваливаются» фронты, уменьшается амплитуда, появляются «наложения» из-за влияния внешних помех, отражений и т. д. Для восстановления формы сигнала используются ретрансляторы, в простейшем случае — обычные логические элементы ТТЛ с повышенной нагрузочной способностью, и желательно с гистерезисной передаточной характеристикой вход — выход. Сигнал с входа А (рис. 3.17, *а*) проходит через элемент D1, линию связи L1, элемент-ретранслятор D2, линию связи L2, выходной элемент D3 и поступает на

Рис. 3.17. Схемы ретрансляции сигналов:

а — использование магистральных логических элементов ТТЛ для однонаправленной передачи сигналов; *б* — неудачная попытка использования двух встречно включенных логических элементов для двунаправленной ретрансляции сигналов; *в* — неудачная попытка использования управляющего сигнала Y

выход В. Резисторы, включенные на концах отрезков линий связи, служат нагрузками для выходных элементов с открытым коллектором и предотвращают отражения сигналов от концов линии. Обычно вместо отдельных резисторов используют резисторные делители напряжения, например 180/390 Ом, см. § 4.4 (этот прием позволяет соблюдать условия согласования линии при меньших энергетических затратах).

При однонаправленной передаче сигналов допустимо включение согласующего резистора или резисторного делителя напряжения только со стороны входа элемента — приемника. Однако в этом случае на коллекторе передающего транзистора могут наблюдаться положительные выбросы напряжения, тем большие, чем хуже выполнены условия согласования линии на приемном конце, а именно чем ближе эти условия к короткому замыканию. Так что, может быть, лучше перестраховаться и согласовать однонаправленную линию с двух концов, а не с одного.

Ретрансляция двунаправленных сигналов, к сожалению, не может производиться столь же простыми средствами. Действительно,

если включить два логических элемента параллельно, но навстречу друг другу, то они образуют триггер (рис. 3.17, *б*), который однажды оказавшись в состоянии, при котором A=B=0, будет оставаться в нем неограниченно долгое время, вплоть до момента снятия напряжения питания.

Использование управляющего сигнала Y (рис. 3.17, *в*), который запрещает передачу информации D в обе стороны одновременно, не решает проблемы, поскольку для передачи сигнала Y нужна дополнительная линия связи, а мы как раз и стремимся сократить их число. Рассмотрим два наиболее экономичных варианта схем двунаправленных приемопередатчиков, не требующих внешнего управляющего сигнала для указания направления передачи.

На рис. 3.18 показана одна из таких схем [33]. В исходном состоянии в точках A и B поддерживаются высокие (+3,5 В) уровни

Рис. 3.18. Двунаправленный ретранслятор (первый вариант)

напряжения, на выходах элементов ИЛИ—НЕ сформированы лог. 0, инверторы с открытым коллектором, выполняющие роль передатчиков ретранслятора, выключены. Отрицательный фронт импульса с входа A после инвертирования поступает на входы элемента И и затем через выходной инвертор проходит на выход B. Цепь R1C1 в данном случае не выполняет полезных функций и вносит нежелательную задержку при прохождении сигнала с входа A на выход B.

При окончании отрицательного импульса в точке A (в точке A — напряжение высокого уровня) сигнал, минуя цепь R1C1, проходит через элемент И и инвертор, вызывая повышение напряжения в точке B до +3,5 В. Поскольку задержка передачи сигнала через выходной инвертор, подключенный к точке B, не равна нулю и сильно зависит от паразитной емкости выхода, при передаче среза импульса возникает ситуация, при которой на входе и выходе этого инвертора одновременно присутствуют лог. 0. Это приводит к появлению кратковременного положительного импульса на выходе элемента ИЛИ — НЕ цепи обратной передачи сигнала.

Если не принять мер по его подавлению, то сработает инвертор, подключенный к точке A, и затем импульс помехи будет распространяться по кольцу, т. е. возникнет генерация. Подавление помехи производится цепью R2C2. Ее параметры выбираются так, чтобы к моменту окончания импульса на верхнем входе элемента И конден-

сатор С2 не успел зарядиться до порогового напряжения этого элемента.

Передача полезного сигнала (отрицательного импульса) в противоположном направлении от В к А производится аналогично. В этом случае помеха на срезе подавляется цепью R1C1. Недостаток схемы поэтому заключается в необходимости подбора RC-цепей для каждого конкретного случая ее применения.

Более совершенной (но и более сложной) является схема, приведенная на рис. 3.19 [34].

По отрицательному фронту сигнала на входе А триггер D1 устанавливается в 1, открывается элемент D2 и сигнал передается через элементы D3 и D4 в канал В. Сигнал лог. 0 с выхода элемента D2

Рис. 3.19. Двунаправленный ретранслятор (второй вариант)

поддерживает триггер D5 в нулевом состоянии, размыкая обратную связь, поэтому по окончании импульса никаких конфликтов не возникает. Если триггер D1 первоначально уже был установлен в 1, то задержка передачи сигнала уменьшается на время срабатывания триггера. Это при многократной ретрансляции может привести к заметным вариациям длительности передаваемых импульсов.

Выходные элементы ретранслятора выполнены по схеме с открытым коллектором; резисторные делители служат в качестве нагрузок для выходных инверторов и обеспечивают согласование с кабелями по волновым сопротивлениям. Передача отрицательного импульса из В в А производится аналогично.

Схему рис. 3.19 можно усовершенствовать с тем, чтобы ее задержка не зависела от состояния триггеров D1 и D5 к моменту поступления входного импульса. Для этого можно ввести два элемента задержки в цепи верхних входов элементов И — НЕ. Каждый из этих элементов задержки будет препятствовать преждевременному прохождению сигнала «в обход» соответствующего триггера, т. е. задержка каждого из введенных элементов выбирается равной или большей, чем задержка триггера. Возможны и иные решения. Например, можно устанавливать оба триггера в 1 в момент окончания передачи отрицательного импульса, а также после включения напряжения питания (чтобы условия распространения первого и последующих импульсов были одинаковыми).

Схемы ретрансляторов, приведенные на рис. 3.18 и 3.19, работают «правильно» только при отсутствии конфликтов между импульсами. Иными словами, ретранслятор передает импульсы в одном или разных направлениях без искажений, если эти импульсы разделены во времени паузами. Если встречные импульсы частично накладываются друг на друга (это обычная ситуация для линии, реализующей функцию Монтажное И (ИЛИ)), то работа схем оказывается довольно своеобразной, если не сказать «неправильной», так как практически любое «своеобразие» можно с пользой применить.

На рис. 3.20 и 3.21 приведены временные диаграммы, соответствующие работе схем при наложении импульсов. Передатчики, подключенные к линии, выполнены по схеме с открытым коллектором.

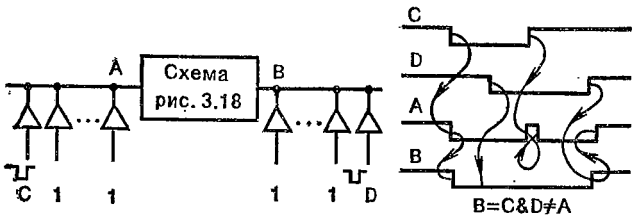


Рис. 3.20. Работа схемы (рис. 3.18) при наложении сигналов C и D:

Функция Монтажное И (C&D) «в чистом виде» реализуется только на отрезке линии, связанном с точкой B, так как в данном примере сигнал D заканчивается после окончания сигнала C

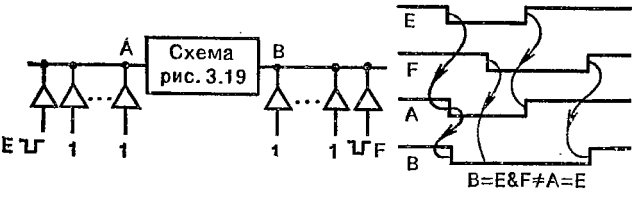


Рис. 3.21. Работа схемы (рис. 3.19) при наложении сигналов E и F.

«Двунаправленность» теряется еще в большей степени, чем при использовании ретранслятора, приведенного на рис. 3.18

Отрезки линий согласованы на концах (согласующие резисторы не показаны). Обе схемы (см. рис. 3.18 и 3.19) не гарантируют правильную реализацию функции Монтажное И, если заранее не известна последовательность формирования встречных импульсов (а так чаще всего и бывает). Еще более причудливые ситуации возникают при рассмотрении цепей из нескольких ретрансляторов.

Таким образом, проблема построения системы ретрансляторов, функционально эквивалентной обычному проводу, не так проста, как это может показаться на первый взгляд. Отметим, что, например, в телефонной связи эта проблема успешно решена еще с появлением

ламповых усилителей, а функция Монтажное ИЛИ реализуется без самовозбуждения системы, в чем мы часто убеждаемся, когда в телефонный разговор вклинивается еще одна (и даже не одна) пара абонентов.

Глава 4

Рекомендации по обеспечению помехозащищенности цифровых устройств

4.1. Подавление помех по первичной питающей сети

Форма сигнала переменного напряжения промышленной питающей сети (~220 В, 50 Гц) в течение коротких промежутков времени может сильно отличаться от синусоидальной — возможны выбросы или «врезки», снижение амплитуды одной или нескольких полуволн и т. д. Причины возникновения таких искажений связаны обычно с резким изменением сетевой нагрузки, например при включении мощного электродвигателя, печи, сварочного аппарата. Поэтому следует по возможности осуществлять развязку от таких источников помех по сети (рис. 4.1).

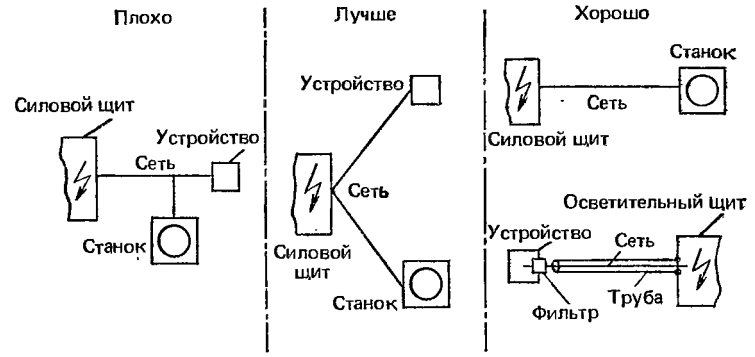


Рис. 4.1. Варианты подключения цифрового устройства к первичной питающей сети

Помимо указаниой меры, возможно, потребуется введение сетевого фильтра на вводе питания устройства с целью подавления кратковременных помех. Резонансная частота фильтра может лежать в пределах 0,15—300 МГц; пирокополосные фильтры обеспечивают подавление помех во всем указанном диапазоне. На рис. 4.2 приведен пример схемы сетевого фильтра Этот фильтр имеет габариты 30×30×20 мм и смонтирован иепосредственно на колодке ввода сети в устройство. В фильтрах должны использоваться высокочастотные

конденсаторы и индуктивности либо без сердечников, либо с высокочастотными сердечниками.

В некоторых случаях обязательным является введение электростатического экрана (обычной водопроводной трубы, соединенной с заземленным корпусом щита питания) для прокладки внутри него проводов первичной питающей сети. Как отмечается в [35], коротковолновый передатчик таксомоторного парка, расположенный на противоположной стороне улицы, способен при определенной взаимной ориентации наводить на отрезке провода сигналы амплитудой несколько сотен вольт. Этот же провод, помещенный в электростатический экран, будет надежно защищен от такого рода наводок.

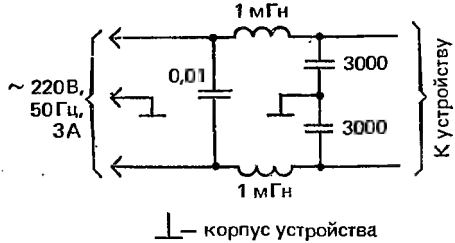


Рис. 4.2. Пример схемы сетевого фильтра

Рассмотрим методы подавления сетевых помех непосредственно в блоке питания устройства. Если первичная и вторичная обмотки силового трансформатора расположены на одной и той же катушке (рис. 4.3, *а*), то за счет емкостной связи между обмотками импульсные помехи могут проходить из первичной цепи во вторичную. Согласно [35] рекомендуются четыре способа подавления таких помех (в порядке возрастания эффективности).

1. Первичная и вторичная обмотки силового трансформатора выполняются на разных катушках (рис. 4.3, *б*). Проходная емкость С уменьшается, однако снижается КПД, так как не весь магнитный поток из области первичной обмотки попадает в область вторичной обмотки из-за рассеяния через окружающее пространство.

2. Первичная и вторичная обмотки выполняются на одной и той же катушке, но разделяются экраном из медной фольги толщиной не менее 0,2 мм. Экран не должен представлять собой короткозамкнутый виток. Он соединяется с корпусной землей устройства (рис. 4.3, *в*).

3. Первичная обмотка полностью заключается в экран, не являющийся короткозамкнутым витком. Экран заземляется (рис. 4.3, *г*).

4. Первичная и вторичная обмотки заключаются в индивидуальные экраны, между которыми прокладывается разделительный экран. Весь трансформатор заключается в металлический корпус (рис. 4.3, *д*). Экраны и корпус заземляются. Этот тип трансформатора в силу предельной защищенности от прохождения помех получил название «ультраизолятор».

При всех перечисленных способах подавления помех разводку сетевых проводов внутри устройства следует выполнять экранированным проводом, соединив экран с корпусной землей. Недопустима ук-

ладка в один жгут сетевых и прочих (питающих платы, сигнальных и т. п.) проводов даже в случае экранирования тех и других.

Рекомендуется параллельно первичной обмотке силового трансформатора в непосредственной близости от выводов обмотки установить конденсатор емкостью примерно 0,1 мкФ и последовательно с ним — токоограничивающий резистор сопротивлением порядка 100 Ом. Это позволяет «замыкать» энергию, накопленную в сердечнике силового трансформатора, в момент размыкания сетевого вы-

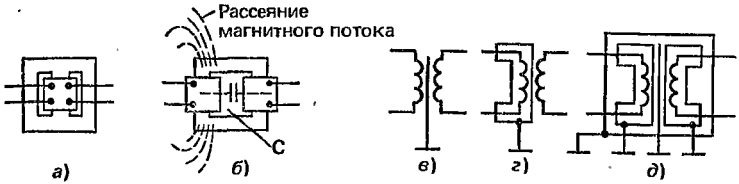


Рис. 4.3. Варианты защиты силового трансформатора от передачи импульсных помех из сети во вторичную цепь (и обратно):

а — защита отсутствует; *б* — разнесение первичной и вторичной обмоток; *в* — прокладка экрана между обмотками; *г* — полная экранировка первичной обмотки; *д* — полная экранировка всех элементов трансформатора

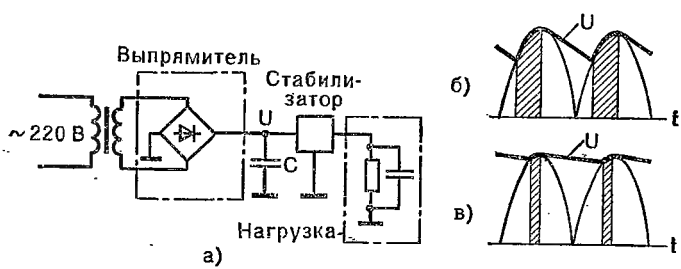


Рис. 4.4. Упрощенная схема блока питания (*а*) и диаграммы (*б*, *в*), поясняющие работу двухполупериодного выпрямителя.

Блок питания является тем бо́льшим источником импульсных помех по сети, чем больше емкость конденсатора С

ключателя. В отсутствие конденсатора эта энергия растекается через паразитные элементы, что может привести к сбоям соседних устройств, связанных с данным по общей сети или (и) по общей земле. Кроме того, уменьшается вероятность сбоев соседних устройств при включении нашего устройства.

Отметим, что с увеличением емкости С фильтра (рис. 4.4, *а*) блока питания нашего устройства увеличивается вероятность сбоев соседних устройств, так как потребление энергии от сети нашим устройством все в большей степени приобретает характер ударов. Действительно, напряжение U на выходе выпрямителя растет в те интервалы времени, когда энергия отбирается от сети (рис. 4.4, *б*). Эти интервалы на рис. 4.4 заштрихованы. С увеличением емкости кон-

денсатора С периоды его заряда становятся все меньшими (рис. 4.4, *в*), а ток, отбираемый в импульсе от сети, — все бо́льшим. Таким образом, внешне «безобидное» устройство может создавать в сети помехи, «не уступающие» помехам от сварочного аппарата.

4.2. Правила заземления, обеспечивающие защиту от помех по «земле»

В устройствах, выполненных в виде конструктивно-законченных блоков, существуют по крайней мере два типа шин «земли» — корпусная и схемная. Корпусная шина согласно требованиям техники безопасности в обязательном порядке подключается к шине заземления, проложенной в помещении. Схемная шина (относительно которой отсчитываются уровни напряжения сигналов) не должна быть

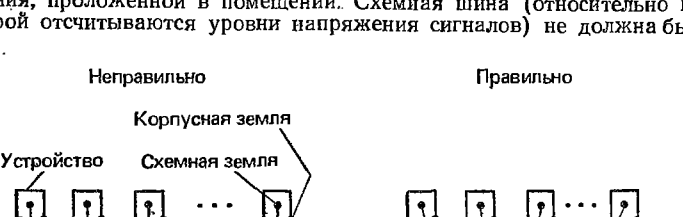


Рис. 4.5. Неправильное и правильное заземление цифровых устройств. Показана шина земли, которая обычно имеется в помещении

соединена с корпусной внутри блока — для нее должен быть выведен отдельный зажим, изолированный от корпуса. На рис. 4.5 показаны варианты неправильного и правильного заземления группы устройств, которые связаны между собой информационными линиями (эти линии не показаны). Схемные шины «земли» объединяются индивидуальными проводами в точке А, а корпусные — в точке В, по возможности приближенной к точке А. Точка А может не подключаться к шине заземления в помещении, однако это создает неудобства, например при работе с осциллографом, у которого «земля» пробника соединена с корпусом.

При неправильном заземлении (см. рис. 4.5) импульсные напряжения, порождаемые уравнивающими токами по земляной шине, будут фактически приложены к входам приемных магистральных элементов, что может вызвать их ложное срабатывание. Следует отметить, что выбор наилучшего варианта заземления зависит от конкретных «местных» условий и зачастую проводится после серии тщательных экспериментов. Однако общее правило (см. рис. 4.5) всегда остается в силе.

4.3. Подавление помех по цепям вторичного электропитания

Моменты переключенин большинства типов интегральных схем из одного состояния в другое сопровождаются резким кратковременным возрастанием тока, потребляемого от источника вторичного электропитания. Энергия, отбираемая от источника питания в эти моменты, расходуется на заряд паразитных емкостей и на протекание «сквозного» тока через выходные каскады, построенные по двухтактной схеме (когда «верхний» и «нижний» транзисторы одновременно находятся в активном режиме). Разряд паразитных выходных емкостей сопровождается появлением кратковременных импульсных токов по земляным шинам.

Из-за конечной индуктивности шин питания и земли импульсные токи вызывают появление импульсных напряжений как положительной, так и отрицательной полярности, которые приложены между выводами питания и земли микросхем. Если шины питания и земли выполнены тонкими печатными или иными проводниками, а высокочастотные развязывающие конденсаторы либо вовсе отсутствуют, либо их число недостаточно, то при одновременном переключении нескольких ТТЛ-микросхем на «дальнем» конце печатной платы амплитуда импульсных помех по питанию (выбросов напряжения, действующих между выводом питания и земли микросхемы) может составить 2 В и более. Поэтому при проектировании печатной платы необходимо выполнять следующие рекомендации.

1. Шины питания и земли должны обладать минимальной индуктивностью. Для этого они выполняются в виде решетчатых структур, покрывающих всю площадь печатной платы. Недопустимо подключение микросхем ТТЛ к шине, представляющей собой «отросток», поскольку по мере приближения к его концу индуктивность цепей подвода питания накапливается. Шины питания и земли должны по возможности покрывать всю свободную площадь печатной платы. С особым вниманием следует подходить к проектированию накопительных матриц динамической памяти на микросхемах К565РУ5, РУ7 и др. Матрица должна представлять собой квадрат, чтобы адресные и управляющие линии имели минимальную длину. Каждая микросхема должна находиться в индивидуальной ячейке решетчатой структуры, образованной шинами питания и земли (две независимые решетки). Шины питания и земли накопительной матрицы не должны нагружаться «чужими» токами, текущими от адресных формирователей, усилителей сигналов управления и т. п.

2. Подключение внешних шин питания и земли к плате через разъем должно производиться через несколько контактов, равномерно расположенных по длине разъема, для того чтобы вход в решетчатые структуры шин питания и земли производился сразу из нескольких точек.

3. Подавление помех по питанию должно осуществляться вблизи мест их возникновения. Поэтому вблизи выводов питания каждой микросхемы ТТЛ должен быть расположен высокочастотный конденсатор емкостью не менее 0,02 мкФ. Это также в особой степени относится к упомянутым микросхемам динамической памяти. Для фильтрации низкочастотных помех необходимо использовать электролитические конденсаторы, например, емкостью 100 мкФ. При использовании микросхем динамической памяти электролитические конден-

саторы устанавливаются, например, по углам накопительной матрицы или в другом месте, но вблизи этих микросхем.

Согласно [36, 37] вместо высокочастотных конденсаторов применяют специальные шины питания BUS-BAR, CAP-BUS, которые прокладывают под линейками микросхем или между ними, не нарушая обычной автоматизированной технологии установки элементов на плату с последующей пайкой «волной». Эти шины представляют собой распределенные конденсаторы с погонной емкостью примерно 0,02 мкФ/см. При той же суммарной емкости, что и при использовании дискретных конденсаторов, шины обеспечивают значительно лучшее подавление помех при более высокой плотности монтажа.

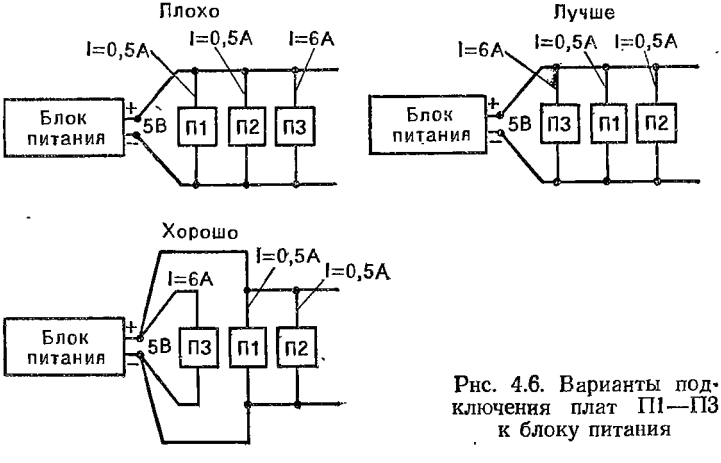


Рис. 4.6. Варианты подключения плат П1—П3 к блоку питания

На рис. 4.6 приведены рекомендации по подключению устройств, выполненных на печатных платах П1—П3, к выходу блока питания. Сильноточное устройство, выполненное на плате П3, создает на шинах питания и земли больше помех, поэтому его следует физически приблизить к блоку питания, а еще лучше обеспечить его питание с помощью индивидуальных шин.

4.4. Правила работы с согласованными линиями связи

На рис. 4.7 показана форма сигналов, передаваемых по кабелю, в зависимости от соотношения сопротивления нагрузочного резистора R и волнового сопротивления кабеля ρ. Сигналы передаются без искажений при R=ρ. Волновое сопротивление конкретного типа коаксиального кабеля известно (например, 50, 75, 100 Ом). Волновое сопротивление плоских кабелей и витых пар обычно близко 110—130 Ом; точное его значение может быть получено экспериментально подбором резистора R, при подключении которого искажения минимальны (см. рис. 4.7). При проведении эксперимента не следует использовать, проволочные переменные сопротивления, так как они

имеют большую индуктивность и могут внести искажения формы сигнала.

Линия связи типа «открытый коллектор» (рис. 4.8). Для передачи каждого магистрального сигнала с длительностью фронта около 10 нс при расстояниях, превышающих 30 см, используется отдельная витая пара или выделяется одна пара жил в плоском кабеле. В пассивном состоянии все передатчики выключены. При срабатывании любого передатчика или группы передатчиков напряжение на линии снижается от уровня, превышающего 3 В, примерно до 0,4 В.

При длине линии 15 м и при правильном ее согласовании длительность переходных процессов в ней не превышает 75 нс. Линия реализует функцию Монтажное ИЛИ по отношению к сигналам, представленным низкими уровнями напряжения.

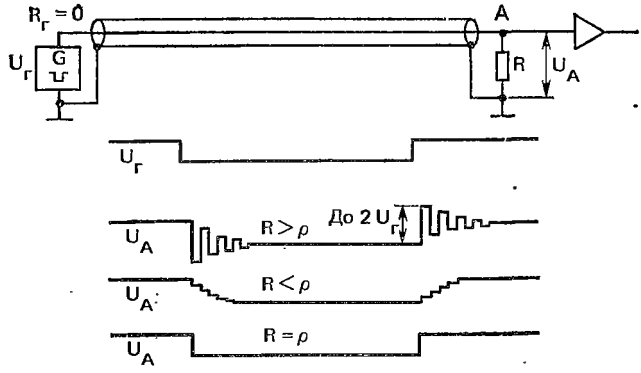


Рис. 4.7. Передача сигналов по кабелю.
G — генератор импульсов напряжения

Линия связи типа «открытый эмиттер» (рис. 4.9). В данном примере показан вариант линии, использующей плоский кабель. Сигнальные провода чередуются с земляными. В идеальном случае каждый сигнальный провод окаймляется с обеих сторон «своими» земляными проводами, однако в этом, как правило, нет особой необходимости. На рис. 4.9 с каждым сигнальным проводом соседствует «своя» и «чужая» земля, что обычно вполне допустимо. Плоский кабель и набор витых пар—по сути почти одно и то же, и все-таки второе предпочтительно в условиях повышенного уровня внешних помех. Линия типа «открытый эмиттер» реализует функцию Монтажное ИЛИ по отношению к сигналам, представленным высокими уровнями напряжения. Временные характеристики приблизительно соответствуют характеристикам линии с «открытым коллектором».

Линия связи типа «дифференциальная пара» (рис. 4.10). Линия применяется для однонаправленной передачи сигналов и характеризуется повышенной помехоустойчивостью, так как приемник реагирует на разность сигналов, а наводимая извне помеха действует на оба провода примерно одинаково. Длина линии практически ограничи-

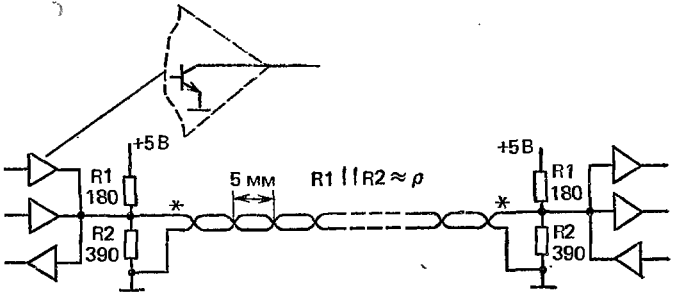


Рис. 4.8. Линия связи типа «открытый коллектор»

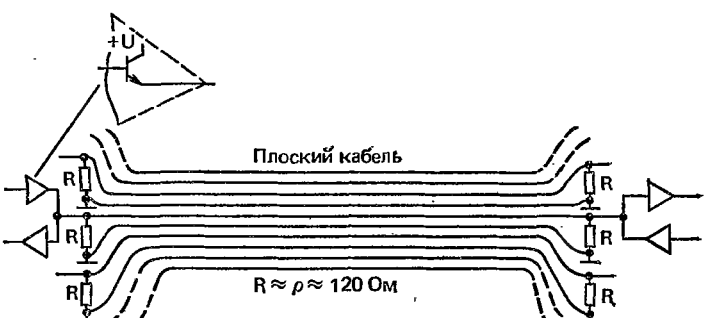


Рис. 4.9. Линия связи типа «открытый эмиттер»

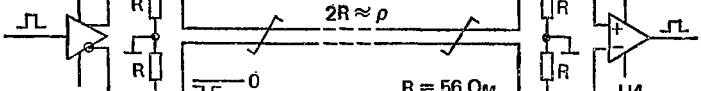


Рис. 4.10. Линия связи типа «дифференциальная пара»

вается омическим сопротивлением проводов и может достигать нескольких сотен метров.

Во всех рассмотренных линиях должны использоваться приемники с высоким входным сопротивлением, малой входной емкостью и предпочтительно с гистерезисной передаточной характеристикой для увеличения помехозащищенности.

Физическая реализация магистрали (рис. 4.11). Каждое устрой-

ство, подключаемое к магистрали, содержит два разъема. Схема, подобная приведенной на рис. 4.11, рассматривалась ранее (см. рис. 3.3), поэтому остановимся лишь на правилах, которые нужно соблюдать при проектировании согласующих блоков (СБ).

1. Каждому резисторному делителю должен соответствовать высокочастотный конденсатор емкостью не менее 0,02 мкФ. Этот конденсатор устанавливается в непосредственной близости от «своего» делителя между шинами питания и земли. Параллельно этим конденсаторам устанавливается электролитический конденсатор для фильтрации низкочастотных помех. Это же правило остается в силе при

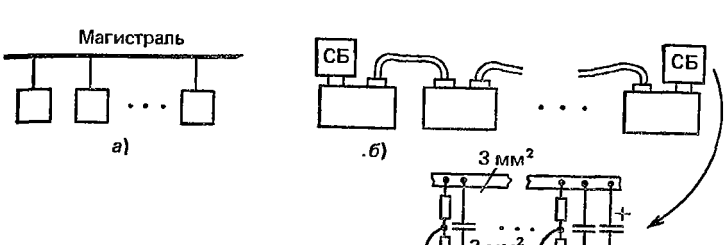


Рис. 4.11. Магистраль *(а)* и ее физическая реализация *(б)* при использовании конструктивно-законченных устройств.
Согласующие блоки подключаются к концам магистрали

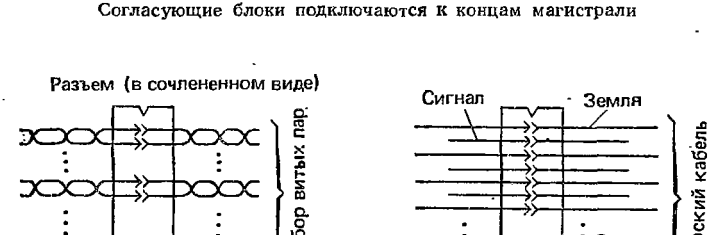


Рис. 4.12. Наилучшие варианты передачи сигналов через разъемы

использовании резисторных сборок — вблизи выводов питания и земли каждой сборки устанавливается высокочастотный конденсатор емкостью не менее 0,1 мкФ.

2. Шины питания и земли должны обладать минимальной индуктивностью, для чего они выполняются в виде «мощных» проводников, например в виде навесных медных планок сечением 3 мм².

3. Несмотря на «общность» земли в согласующем блоке (СБ), к каждому земляному выводу резисторных делителей должна подходить «своя» земля из кабеля. Другими словами, если, например, сигнальный провод витой пары приходит на среднюю точку некоторого

резисторного делителя, то второй ее провод должен быть припаян к земляному выводу именно этого делителя.

Передача магистральных сигналов через разъемы. Наилучшие варианты распайки разъемов показаны на рис. 4.12. Фронт бегущего по магистрали импульса в этих случаях почти «не чувствует» разъема, так как вносимая в кабельную линию неоднородность незначительна. При этом, однако, требуется занять 50 % используемых контактов под земли.

Если это условие по каким-либо причинам невыполнимо, то можно в ущерб помехозащищенности принять второй, более экономичный по числу контактов вариант распайки разъемов, показанный на рис. 4.13. Этот вариант часто используется на практике. Земли витых пар (или земли плоского кабеля) собираются на металлические планки по возможности большего сечения, например 5 мм². Распайка этих земель ведется равномерно по длине планки, по мере распайки соответствующих сигнальных проводов. Обе планки объединяются че-

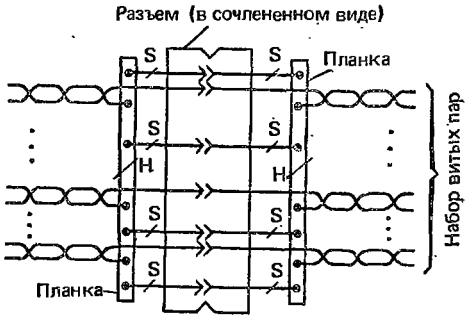


Рис. 4.13. Допустимый вариант передачи сигналов через разъем.
H=5 мм² — сечение планки, S=0,5 мм² — сечение земляного провода

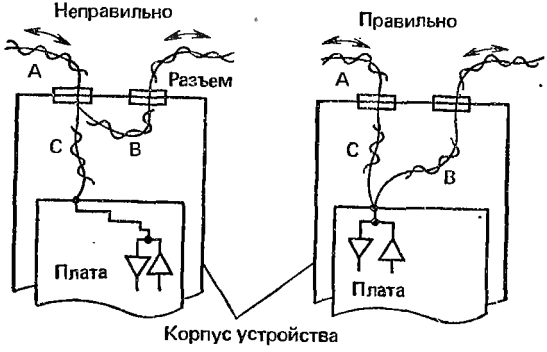


Рис. 4.14. Варианты выполнения ответвлений от магистрали

рез разъем с помощью ряда перемычек минимальной длины и максимального сечения, причем перемычки располагаются равномерно по длине планок. Каждая земляная перемычка должна соответствовать не более чем четырем сигнальным линиям, но общее число перемычек не должно быть меньше трех (одна в центре и две по краям).

Выполнение ответвлений от магистрали. На рис. 4.14 показаны варианты неправильного и правильного выполнения ответвления от магистрали. Прослежен путь одной линии, земляной провод показан условно. Первый вариант (типичная ошибка начинающих схемотехников!) характеризуется расщеплением на две части энергии волны,

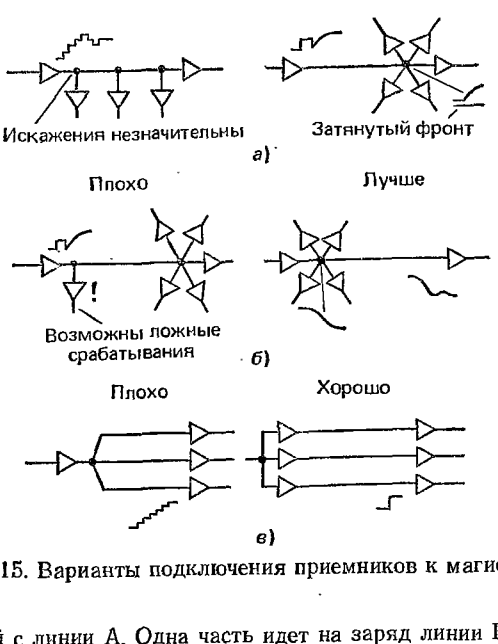


Рис. 4.15. Варианты подключения приемников к магистрали

приходящей с линии А. Одна часть идет на заряд линии В, другая — на заряд линии С. После заряда линии С «полноценная» волна начинает распространяться по линии В, пытаясь догнать ушедшую ранее волну с половинной энергией. Фронт сигнала, таким образом, имеет ступенчатую форму. При правильном выполнении ответвления отрезки линий А, С и В оказываются включенными последовательно, поэтому волна практически не расщепляется и фронты сигналов не искажаются. Передатчики и приемники, расположенные на плате, должны быть максимально приближены к ее краю для уменьшения неоднородности, вносимой в точку объединения отрезков линий В и С.

На рис. 4.15 приведены дополнительные рекомендации по работе с кабельными линиями [38]. Приемники лучше располагать равномерно по магистрали, чем собирать их в «пучок»: пусть лучше будет много мелких неоднородностей, чем одна большая (рис. 4.15, *а*).

Если приемники не удается рассредоточить, то лучше расположить их в начале линии, чтобы сразу «завалить» фронты сигналов паразитной емкостью пучка. (Сигналы с пологими фронтами менее подвержены искажениям при передаче по линии.) Особую опасность при наличии пучка в конце линии представляет включение отдельного приемника в ее начале, где сигнал сильно искажен (рис. 4.15, *б*).

Для развязки пучков приемников от магистрали можно использовать одно или двунаправленные приемопередатчики (см. рис. 3.18, 3.19). При разветвлении линии на несколько направлений для каждого следует выделить отдельный передатчик (рис. 4.15, *в*).

Для передачи по линии лучше использовать не прямоугольные, а трапецеидальные импульсы [39]. Сигналы с пологими фронтами, как отмечалось, распространяются вдоль линии с меньшими искажениями. В принципе в отсутствие внешних помех для любой сколь угодно длинной и даже несогласованной линии можно подобрать настолько медленную скорость нарастания сигнала, что передаваемый и принимаемый сигналы будут отличаться на сколь угодно малую величину. Для получения трапецеидальных импульсов передатчик выполняется в виде дифференциального усилителя с интегрирующей цепью обратной связи. На входе магистрального приемника, выполненного также в виде дифференциального усилителя, устанавливается интегрирующая цепь для фильтрации высокочастотных помех.

При передаче сигналов в пределах платы, когда число приемников велико, часто используют «последовательное согласование». Оно состоит в том, что последовательно с выходом передатчика, в непосредственной близости от этого выхода, включается резистор сопротивлением 20—50 Ом. Это позволяет погасить колебательные процессы на фронтах сигналов. Такой прием часто используют при передаче сигналов управления (RAS, CAS, WE) от усилителей к БИС динамической памяти.

4.5. О защитных свойствах кабелей [40]

На рис. 4.16, *а* показана простейшая схема передачи сигналов по коаксиальному кабелю, которая в ряде случаев может считаться вполне удовлетворительной. Ее основной недостаток состоит в том, что при наличии импульсных уравнивающих токов между корпусными землями (уравнивание потенциалов — основная фуикция системы корпусных земель) часть этих токов i может течь по оплетке кабеля и вызывать падение напряжения (в основном из-за индуктивности оплетки), которое в конечном счете действует на нагрузку R. Вследствие этого число точек соединения оплетки кабеля с корпусными землями следует сводить к минимуму, в частности к нулю (рис. 4.16, б). Эта мера, однако, не спасает от возникновения импульсных (или высокочастотных) токов в оплетке из-за связи с источником помех через паразитную емкость между источником помех и оплеткой. Более того, в этом смысле схема, приведенная на рис. 4.16, *а*, оказывается предпочтительной, и с увеличением числа точек соприкосновения оплетки кабеля с корпусной землей улучшаются возможности стекания наведенных зарядов с оплетки. Использование кабеля с дополнительной оплеткой (рис. 4.16, *в*) позволяет защититься как от емкостных наводок, так и от уравнивающих токов, которые в этом случае текут по внешней оплетке и практически не влияют на сигнальную цепь.

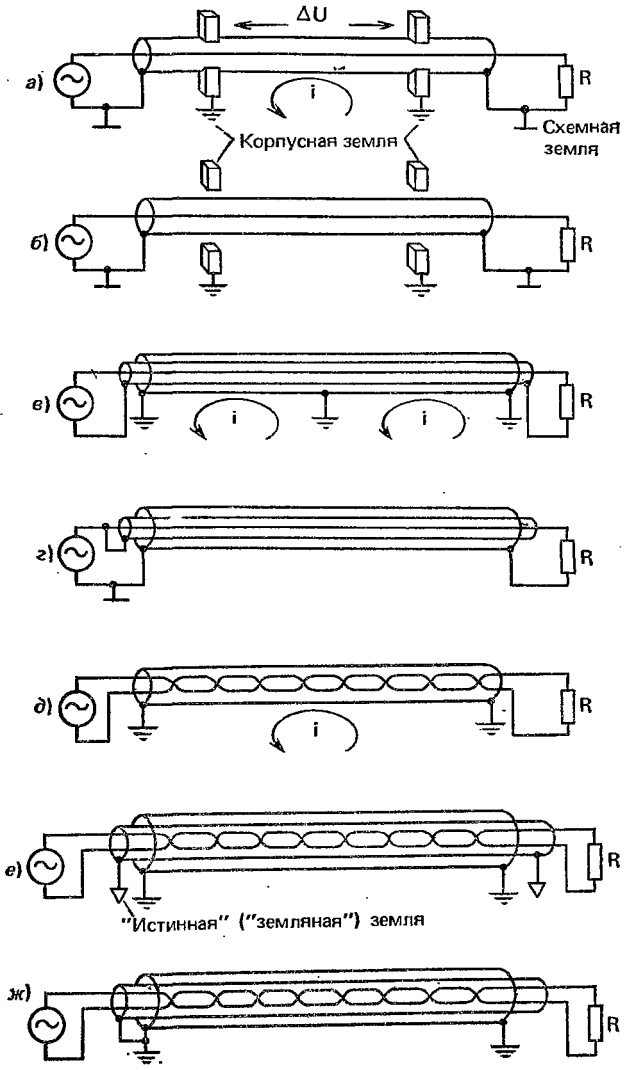


Рис. 4.16. Варианты использования кабелей

Включение кабеля с дополнительной оплеткой по схеме, показанной на рис. 4.16, *г*, позволяет улучшить частотные свойства линии путем уменьшения ее погонной емкости. В идеальном случае потенциал любого элементарного участка центральной жилы совпадает с потенциалом элементарного цилиндра внутренней оплетки, окружающего этот участок. Если это так, то элементарная емкость между указанными проводниками никогда не перезаряжается, так как потенциалы на обеих ее обкладках меняются синфазно. Поэтому можно считать, что паразитная емкость между центральной жилой и внутренней оплеткой равна нулю. Энергия на перезаряд емкости между оплетками отбирается непосредственно от источника сигналов, а энергия, подаваемая на центральную жилу, не расходуется «понапрасну» (на заряд линии, как в обычном коаксиальном кабеле) и полностью выделяется на нагрузке. Линии такого типа используются в локальных сетях ЭВМ для повышения скорости передачи информации. Внешняя оплетка кабеля является частью сигнальной цепи, и поэтому данная схема с точки зрения защищенности от внешних помех эквивалентна схеме, показанной на рис. 4.16, *б*.

Ни медная, ни алюминиевая оплетка простого коаксиального кабеля не защищает его от воздействия низкочастотных магнитных полей. Эти поля наводят ЭДС как на отрезке оплетки, так и на соответствующем отрезке центральной жилы. Хотя эти ЭДС и одноименны по знаку, они не компенсируют друг друга по величине из-за разной геометрии соответствующих проводников — центральной жилы и оплетки. Разностная ЭДС в конечном счете прикладывается к нагрузке R. Дополнительная оплетка (рис. 4.16, *в*, *г*) также не способна предотвратить проникновение магнитного поли низкой частоты в ее внутреннюю область.

Защиту от низкочастотных магнитных полей обеспечивает кабель, содержащий витую пару проводов, заключенную в оплетку (рис. 4.16, *д*). В этом случае ЭДС, наводимые внешним магнитным полем на составляющих витую пару проводах, полностью компенсируют друг друга как по знаку, так и по абсолютной величине. Это тем более справедливо, чем меньше шаг свивания проводов по сравнению с зоной действия поля и чем более тщательно (симметрично) выполнена скрутка. Недостатком такой линии является ее сравнительно низкий частотный «потолок» — порядка 15 МГц — из-за больших потерь энергии полезного сигнала на более высоких частотах.

Схема, представленная на рис. 4.16, *е*, обеспечивает наилучшую защиту от всех видов помех (емкостные наводки, уравнивающие токи, низкочастотные магнитные поля, высокочастотные электромагнитные поля). Внутреннюю оплетку рекомендуется соединять с «радиотехнической» или «истинной» (в прямом смысле — заземленной) землей, а внешнюю — с «системной» (схемной или корпусной) землей. При отсутствии «истинной» земли можно воспользоваться схемой включения, показанной на рис. 4.16, *ж*. Внешняя оплетка соединяется с системной землей на обоих концах, а внутреняя — только со стороны источника. В тех случаях, когда нет необходимости в защите от низкочастотных магнитных полей и есть возможность передавать информацию без использования парафазных сигналов, один из проводов витой пары может служить сигнальным проводом, а второй — экраном. В этих случаях схемы, приведенные на рис. 4.16, *е*, *ж*, можно рассматривать как коаксиальные кабели с тремя экранами — земляным проводом витой пары, внутренней и внешней оплетками кабеля.

4.6. Использование оптронных развязок для подавления помех

Если устройства системы разнесены на значительное расстояние, например на 500 м, то трудно рассчитывать на то, что их земли всегда имеют один и тот же потенциал. Как отмечалось, уравнивающие токи по земляным проводникам создают импульсные помехи на этих проводниках за счет их индуктивности. Эти помехи в конечном счете прикладываются к входам приемников и могут вызвать их ложное срабатывание. Использование линий типа «дифференциальная пара» (см. § 4.4) позволяет подавлять лишь синфазные помехи и поэтому не всегда дает положительные результаты. На рис. 4.17 показаны схемы оптронных развязок между двумя удаленными друг от друга устройствами.

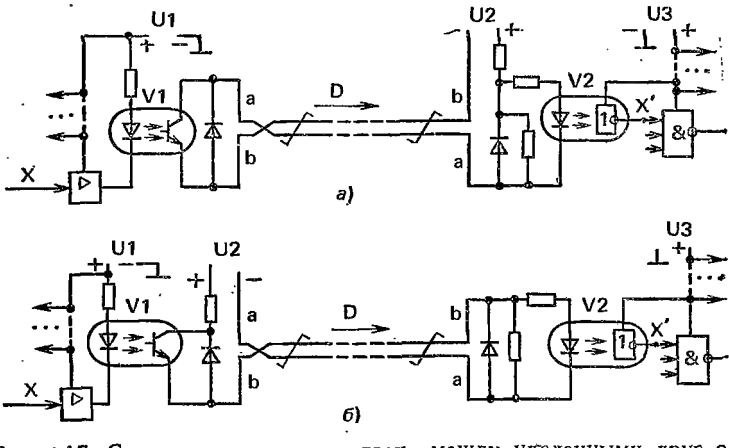


Рис. 4.17. Схемы оптронных развязок между удаленными друг от друга устройствами:

а — с активным приемником, *б* — с активным передатчиком

Схема с «активным приемником» (рис. 4.17, *а*) содержит передающий оптрон V1 и приемный оптрон V2. При подаче импульсных сигналов на вход X светодиод оптрона V1 периодически излучает свет, в результате выходной транзистор этого оптрона периодически насыщается и сопротивление между точками a и b падает от нескольких сотен килоом до нескольких десятков ом. При включении выходного транзистора передающего оптрона ток от положительного полюса источника U2 проходит через светодиод оптрона V2, линию (точки a и b) и возвращается к отрицательному полюсу этого источника. Источник U2 выполняется изолированным от источника U3. Если выходной транзистор передающего оптрона выключен, то ток по цепи источника U2 не протекает. Сигнал X′ на выходе оптрона V2 близок к нулю, если его светодиод включен, и близок +4 В, если этот светодиод выключен. Таким образом, при X=0 светодиоды

передающего и приемного оптронов включены и, следовательно, X′=0. При X=1 оба светодиода выключены и X′=1.

Оптронная развязка позволяет значительно повысить помехоустойчивость канала связи и обеспечить передачу информации на расстояния порядка сотен метров. Диоды, подключенные к передающему и приемному оптронам, служат дли их защиты от обратных выбросов напряжения. Резисторная цепь, связанная с источником U2, служит для задания тока в линии и ограничения тока через светодиод приемного оптрона. Ток в линии согласно интерфейсу ИРПС может быть выбран равным 20 или 40 мА. При выборе номиналов резисторов нужно учитывать омическое сопротивление линии связи. Схема с «активным передатчиком» (рис. 4.17, *б*) отличается от предыдущей тем, что источник питания линии U2 расположеи на стороне передатчика. Это не дает никаких преимуществ — обе схемы по сути одинаковы и являются так называемыми «токовыми петлями».

Рекомендации, приведенные в этой главе, могут показаться начинающему схемотехнику слишком жесткими. Борьба с помехами представляется ему «сражением с ветряной мельницей», а отсутствие опыта работы по проектированию устройств повышенной сложности создает иллюзию того, что можно создать работоспособное устройство, не выполнив ни одной из приведенных рекомендаций. Действительно, иногда возможно и такое. Известны даже случаи серийного выпуска таких устройств. Однако в неофициальных отзывах об их работе можно услышать много интересных нетехнических выражений, таких, как *визит-эффект* и некоторых других, более простых и понятных.

РАЗДЕЛ II

ПРИМЕРЫ ПОСТРОЕНИЯ МИКРОПРОЦЕССОРНЫХ СИСТЕМ И ИХ СОСТАВНЫХ ЧАСТЕЙ

Глава 5

Взаимодействие с объектами

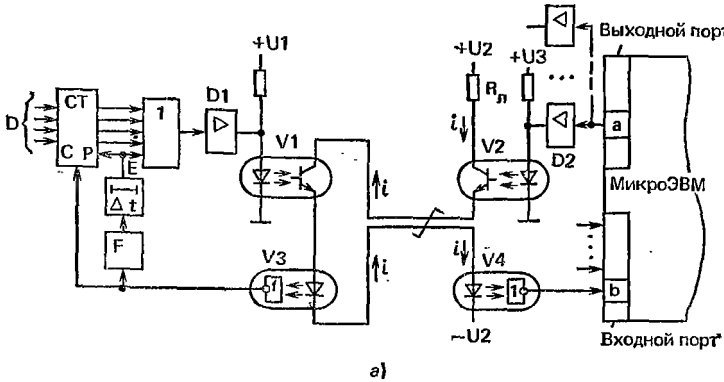
5.1. Опрос удаленного источника информации по двухпроводной линии связи

Первый вариант схемного решения [112]. Задача состоит в том, чтобы микроЭВМ могла по собственной инициативе считывать информацию с линий D, используя последовательный канал связи и асинхронный принцип обмена, без жесткой привязки к какой-либо синхронизирующей сетке (рис. 5.1, *а*).

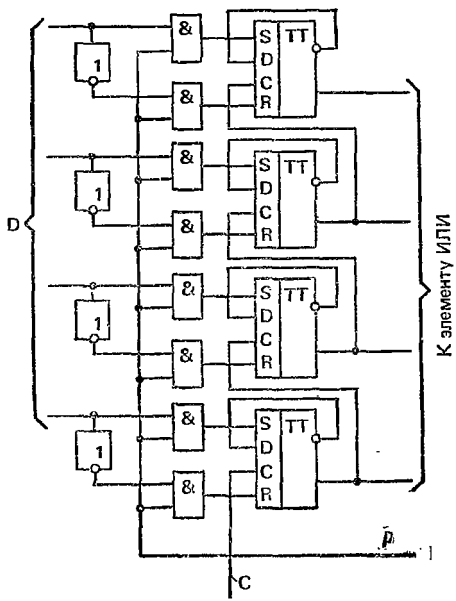
Счетчик СТ при E=0 нечувствителен к сигналам на входах D и работает в режиме вычитания единиц из старого содержимого при поступлении положительных фронтов сигнала на вход C. При E=1 счетчик нечувствителен к сигналам на входе C. При переходе сигнала E из 1 в 0 в счетчике фиксируется текущее состояние сигналов на линиях D. Пример выполнения счетчика приведен на рис. 5.1, *б*. Одновибратор F вырабатывает на выходе отрицательный импульс при воздействии на его вход положительного фронта сигнала. Длительность этого сигнала должна превышать время сеанса связи микроЭВМ с источником информации и может составлять, например, 50 мс. Элемент задержки препятствует одновременному воздействию на счетчик сигнала смены режима и синхросигнала.

В исходном состоянии a=1, элемент D2 с открытым коллектором выключен, транзистор оптрона V2 насыщен благодаря действию светового потока. Поскольку предыдущий сеанс связи кончился, E=1, элемент D1 с открытым коллектором выключен, транзистор оптрона V1 насыщен, в линии течет ток i. Сеанс связи начинается в момент t_0 (рис. 5.2) по инициативе микроЭВМ, которая выдает сигнал a=0, что вызывает прекращение тока в линии. На выходе оптрона V3 формируется первый положительный фронт сигнала, который не воздействует на состояние счетчика.

Предположим сначала, что в момент перехода сигнала E из 1 в 0 в счетчике зарегистрирован нулевой код. В этом случае на выходе элемента ИЛИ формируется лог. 0 и транзистор оптрона V1 выключается, подтверждая разомкнутое состояние линии. В момент t_1 микроЭВМ пытается вызвать протекание тока в линии установкой разряда a в единичное состояние и через интервал времени,



а)



б)

Рис. 5.1. Первый вариант схемного решения устройства для связи микроЭВМ с удаленным источником информации:

а — схема канала связи; *б* — пример выполнения счетчика на D-триггерах

достаточный для окончания переходных процессов в линии, опрашивает состояние входа b. Поскольку оптрон V1 выключен, такая попытка даст отрицательный результат — тока не будет и сигнал b=1 (см. на рис. 5.2 эпюру i тока i при нулевом начальном содержимом счетчика). Таким образом, микроЭВМ узнаёт, что в момент опроса на линиях D присутствовал нулевой код.

Если первоначальное состояние счетчика 0001_2, то первая попытка выдачи тока i в линию будет успешной (см. первый импульс тока на эпюре g, построенной для этого начального состояния). При окончании первого импульса тока (см. точку на его фронте) из старого содержимого счетчика вычитается единица, поэтому, как и в предыдущем примере, на выходе элемента ИЛИ формируется лог. 0 и уже разомкнутая оптроном V2 линия связи дополнительно размыкается оптроном V1. Вторая попытка возбуждения тока в линии в этом случае будет безуспешной, и, обнаружив, что b=1, микроЭВМ узнаёт, что на линиях D был установлен код 0001.

Эпюры c, d и e на рис. 5.2 соответствуют случаям, когда в счетчике первоначально содержатся двоичные коды 0010, 0011 и 1111. Из этих эпюр видно, что число импульсов тока в линии в те-

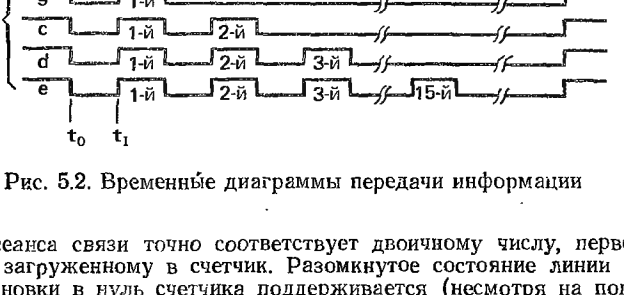


Рис. 5.2. Временные диаграммы передачи информации

чение сеанса связи точно соответствует двоичному числу, первоначально загруженному в счетчик. Разомкнутое состояние линии после установки в нуль счетчика поддерживается (несмотря на попытку микроЭВМ выдать ток) до тех пор, пока сигнал E=0. При E=1 система приходит в описанное ранее исходное состояние.

При работе с несколькими каналами их опрос производится параллельно, с использованием общего разряда a выходного порта. В этом случае микроЭВМ одновременно считывает ответные реакции от всех последовательных каналов (параллельным кодом) и программно анализирует полученное слово. Когда во всех разрядах этого слова зарегистрированы единицы (попытка выдачи тока во все линии оказалась безуспешной), сеанс связи заканчивается.

Схема не предъявляет жестких ограничений на временные параметры передаваемых сигналов, как при использовании, например, «телеграфного» кода. МикроЭВМ может сужать или расширять импульсы тока и (или) паузы между ними в широких пределах. Максимальная скорость обмена ограничена временем «успокоения» сигиалов в линии, а минимальная — длительностью импульса E, которая может быть выбрана с нужным запасом.

Если коды, выдаваемые источником информации, имеют разрядность, равную n, то, как следует из временных диаграмм, представленных на рис. 5.2, в худшем случае (при передаче кода 111...1) сеанс связи будет сопровождаться посылкой 2^n-1 импульсов тока по линии. Поэтому с увеличением n эффективность работы схемы снижается и ее применимость ограничивается случаями, при которых n не превышает 5 или 6.

Простая последовательная передача кода по линии требует не

2^n-1, а n элементарных циклов. Однако рассмотренное далее схемное решение оказывается более сложным и его использование оправдано при бо́льших значениях n, например при n=10.

Второй вариант схемного решения. В отличие от предыдущей схемы, в данном случае передача кода X от удаленного источника информации к микроЭВМ по двухпроводной линии связи L производится путем выполнении серии сдвигов (рис. 5.3). Управление процессом передачи кода возложено на программу, которая оперирует сигналами программно-доступных входного и выходного портов. Эта схема, как и предыдущая, не требует генерации импульсов калиброванной длительности, как, например, в интерфейсе ИРПС. Процесс передачи кода может быть даже временно прерван в любой момент в связи с временным переходом микроЭВМ к выполнению более приоритетного задания. Ограничение сверху на длительность сеанса связи состоит в том, что эта длительность не должна превышать длительности импульса в точке S, который формируется одновибратором D4.

Схема (рис. 5.3) содержит две секции — передающую и приемную. Передающая секция находится на некотором удалении от приемной, например на расстоянии 300 м. Линия связи L между этими секциями выполняется витой парой проводов.

Передающая секция содержит сдвиговый регистр, элемент ИЛИ D1 с открытым коллектором, элемент И D2, одновибратор D4, элемент задержки D3, передающий оптрон V1, приемные оптроны V2 и V3, а также защитные диоды VD1 и VD2, которые служат для предотвращения пробоя светодиода оптрона V3 при воздействии напряжения обратной полярности. Резистор R1 задает ток от источника питания U1 через светодиод оптрона V1 в тех случаях, когда на входах элемента ИЛИ D1 присутствует по крайней мере одна лог. 1.

Сдвиговый регистр работает в двух режимах. При S=1 регистр нечувствителен к сигналам на входе C и в его разрядах устанавливается код X. В момент перехода сигнала S из 1 в 0 код X фиксируется в регистре и в дальнейшем, при поступлении положительных перепадов напряжения по входу C, этот код последовательно сдвигается в регистре в направлении, указанном стрелкой. Пример выполнения сдвигового регистра на D-триггерах приведен на рис. 5.4.

Одновибратор вырабатывает отрицательный импульс при поступлении на его вход положительного перепада напряжения. Элемент задержки D3 препятствует одновременному воздействию на сдвиговый регистр сигнала смены режима и сигнала сдвига.

Приемная секция подключена к выходному и входному портам микроЭВМ и содержит передающие оптроны V4, V6, приемный оптрон V5 и два буферных элемента D5 и D6 с открытым коллектором. Резисторы R3 и R4 задают ток через светодиоды оптронов V4 и V6 при выключенных элементах D5 и D6. Резисторы R2 и R5 предназначены соответственно для задания «положительного» тока i1 и «отрицательного» тока i2 в линии L.

Приемный оптрон V2 (V3, V5) формирует на выходе сигнал лог. 0 при наличии тока через светодиод, при отсутствии тока сигнал равен лог. 1. Транзистор передающего оптрона V1 (V4, V6) насыщается только тогда, когда соответствующий светодиод излучает свет и потенциал коллектора этого транзистора более положителен, чем потенциал эмиттера.

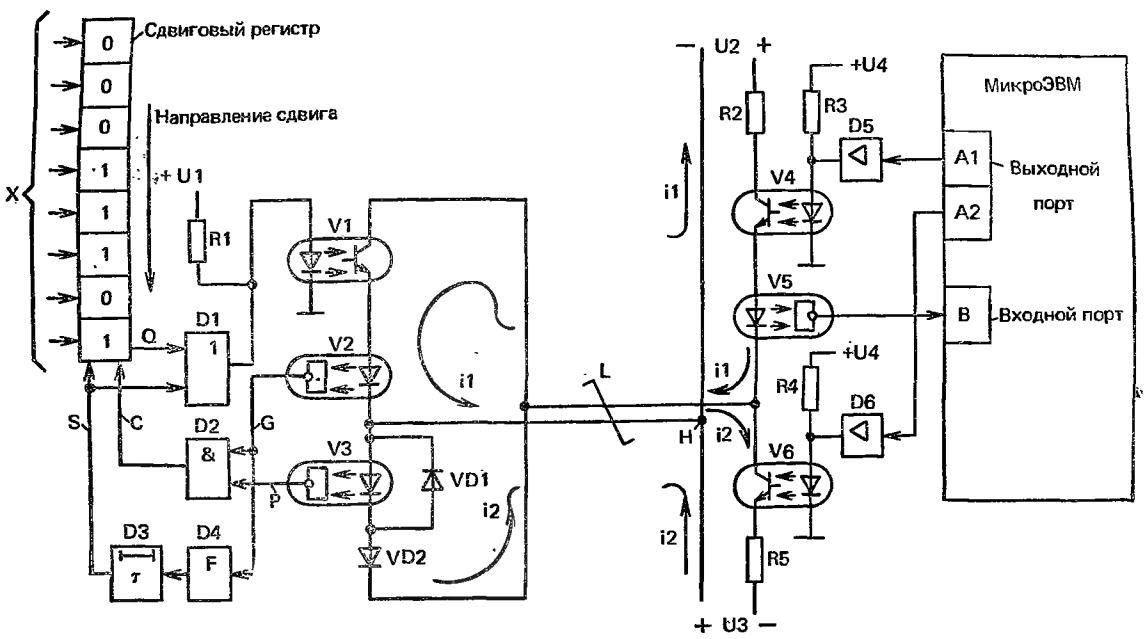


Рис. 5.3. Второй вариант схемного решения устройства для связи микроЭВМ с удаленным источником информации

Источник питания U1 обеспечивает питание части схемы, непосредственно связанной с источником информации. Источники питания U2 и U3 предназначены для задания положительного или отрицательного тока в линии L. Эти источники по существу представляют собой один источник со средней точкой H. Источник питания U4 обеспечивает питание части схемы, связанной с микроЭВМ. Таким образом, линия связи L гальванически развязана с источниками питания U1 и U4, что обеспечивает необходимую помехозащищенность.

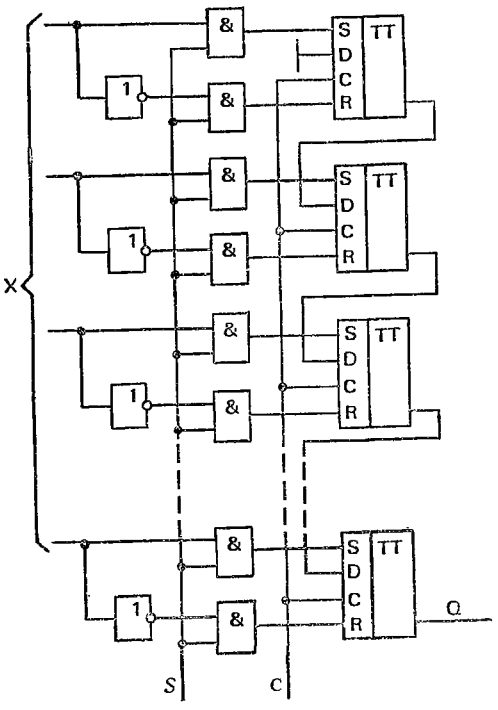


Рис. 5.4. Пример выполнения схемы регистра

В исходном состоянии S=1 (предыдущий сеанс связи уже кончился), поэтому светодиод оптрона V1 излучает свет. Из выходного порта микроЭВМ поступают сигналы A1=1 и A2=0, поэтому светодиод оптрона V4 включен, а светодиод оптрона V6 выключен. При этом по цепи источника U2 протекает ток i1:

(+U2) — R2 — V4 — V5 — L — V1 — V2 — L — (—U2).

Протекание этого тока приводит к формированию в точке G и на входе B сигналов лог. 0. В точке P устанавливается лог. 1 из-за отсутствия тока i2 по цепи источника питания U3.

В момент t_0 микроЭВМ формирует на выходе A1 сигнал лог. 0,

оставляя без изменения сигнал на выходе А2 (рис. 5.5). Это приводит к прекращению тока в линии L, формированию сигнала G=1 и запуску одновибратора. Фронт сигнала С не оказывает воздействия на сдвиговый регистр, поскольку благодаря элементу задержки сигнал S в момент действия фронта равен лог. 1. Задержанный сигнал, выработанный одновибратором, в момент t_1 воздействует на режимный вход сдвигового регистра, подготавливая его к последующим операциям сдвига с поступлением положительных перепадов напряжения на вход С.

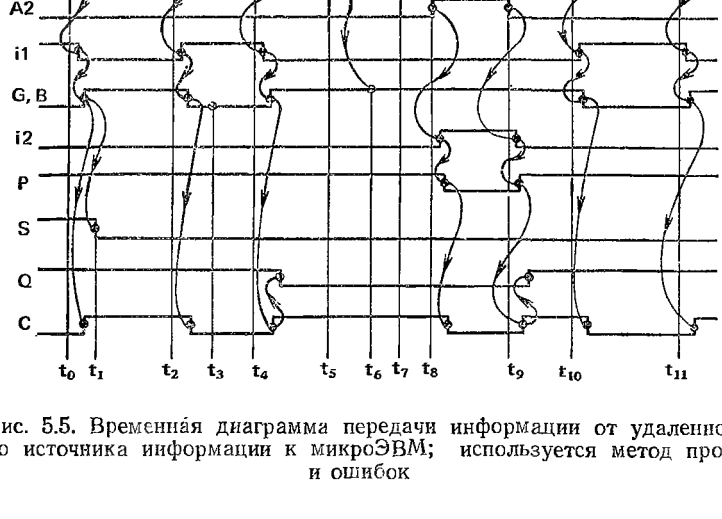


Рис. 5.5. Временная диаграмма передачи информации от удаленного источника информации к микроЭВМ; используется метод проб и ошибок

В примере, приведенном на рис. 5.3, в нижнем разряде сдвигового регистра в момент t_1 зафиксирована лог. 1, поэтому светодиод оптрона V1 остается включенным, несмотря на то, что S=0.

Выждав определенное время, достаточное для окончания переходных процессов в линии и в цепи V2 — D4 — D3 — D1 — V1, микроЭВМ вновь пытается вызвать протекание тока i1 по цепи источника U2 (момент t_2) путем выдачи сигнала лог. 1 в разряд A1 выходного порта с сохранением сигнала лог. 0 в разряде А2. Логика работы здесь такова: если попытка оказалась удачной, то B=0 и микроЭВМ узнаёт, что в нижнем разряде сдвигового регистра присутствует лог. 1; если же ток i1 вызвать не удалось, то это означает, что оптрон V1 оказался выключенным, т. е. в нижнем разряде сдвигового регистра в данный момент присутствует лог. 0.

В данном случае микроЭВМ, выждав для «успокоения» схемы время t_2—t_8, опрашивает входной порт и, поскольку B=0, программа принимает к сведению, что в нижнем разряде сдвигового регистра присутствует лог. 1.

В момент t_4 микроЭВМ выдает в разряд A1 выходного порта сигнал лог. 0. Ток i1 вновь прекращается и на входе С сдвигового

регистра формируется сигнал лог. 1. На этот раз сдвиговый регистр воспринимает фронт импульса С как сигнал сдвига, так как S=0. В нижний разряд сдвигового регистра попадает следующий бит, равный в данном примере 0.

В момент t_5 микроЭВМ вновь пытается вызвать протекание тока i1 установкой лог. 1 в разряде A1 выходного порта, но на этот раз попытка оказывается неудачной, поскольку Q=0, S=0, светодиод оптрона V1 шунтируется сигналом лог. 0 с выхода элемента D1. В момент времени t_6 микроЭВМ опрашивает выходной порт и, восприняв сигнал B=1, узнаёт о том, что после первого сдвига в нижнем разряде сдвигового регистра оказался 0.

Чтобы продолжить сдвиг, микроЭВМ вынуждена воспользоваться «обходной» цепью, создав ток i2 в цепи

(+U3) — L — V3 — VD2 — L — V6 — R5 — (—U3).

В моменты t_7, t_8 и t_9 микроЭВМ меняет коды в разрядах выходного порта, что приводит к формированию «искусственного» отрицательного импульса на входе С сдвигового регистра. В результате в нижний разряд сдвигового регистра попадает 1, которая в период времени t_{10}—t_{11} передается в микроЭВМ, как уже было рассмотрено (см. описание работы схемы в интервале времени t_2—t_4).

Далее процесс ввода продолжается (на рис. 5.5. не показан) до тех пор, пока все биты из сдвигового регистра не окажутся в памяти микроЭВМ. Программа в процессе ввода собирает из принимаемых битов единое слово. После приема всех бит микроЭВМ устанавливает на выходах сигналы A1=1 и A2=0. Через некоторое время заканчивается отрицательный импульс S и схема возвращается в описанное ранее исходное состояние.

Рассмотренные здесь (и далее в этом параграфе) схемы позволяют вычислительной машине узнавать о состоянии входных линий X (рис. 5.3) в момент опроса, т. е. в момент формирования отрицательного перепада напряжения на входе S сдвигового регистра. Насколько «объективна» полученная информация?

Процесс опроса по своей сути схож с фотосъемкой группы позирующих людей — объекта более или менее статичного, но тем не менее подверженного переходным процессам. Зачастую на групповых фотоснимках можно увидеть человека с непреднамеренно закрытыми глазами. Точно так же момент опроса удаленного источника информации машиной может совпасть с моментом смены информации на линиях X. Как распознать такие ситуации? Вспомним, что опытные фотографы делают несколько исходных групповых снимков в расчете на то, что по крайней мере один из них окажется полноценным. Эта же идея «подстраховки» легко воплощается на программном или на аппаратном уровне.

Программная реализация. МикроЭВМ считывает код с линий X два раза подряд и сравнивает полученные результаты. Если результаты не совпали, то это может означать, что:

а) состояние удаленного объекта за время между первым и вторым считыванием успело измениться, так что первый результат соответствует старому состоянию, а второй — новому; оба результата по-своему верны;

б) один результат принципиально неверен (не исключена возможность того, что неверны оба результата), так как в момент «защелкивания» сдвигового регистра по входу S менялся код на линиях X: часть разрядов регистра уже успела перейти в новое

состояние, а другая часть все еще хранит старую информацию; например, в процессе перехода из состояния 111...1 в состояние 000..., ...0 возможно появление любых промежуточных комбинаций.

Можно ли отличить первый случай от второго? Если известны особенности «поведения» удаленного объекта, то это сделать можно. Например, если код на линиях X представляет собой результат непрерывного измерения температуры массивного тела (с точностью до одного градуса), то вряд ли можно ожидать, что результат может измениться более чем на один градус за период времени, равный 2 мс. В этом случае программа проявляет «терпимость» к незначительным отличиям между двумя последовательными результатами считывания, а скачкообразные изменения она расценивает как неправдоподобные.

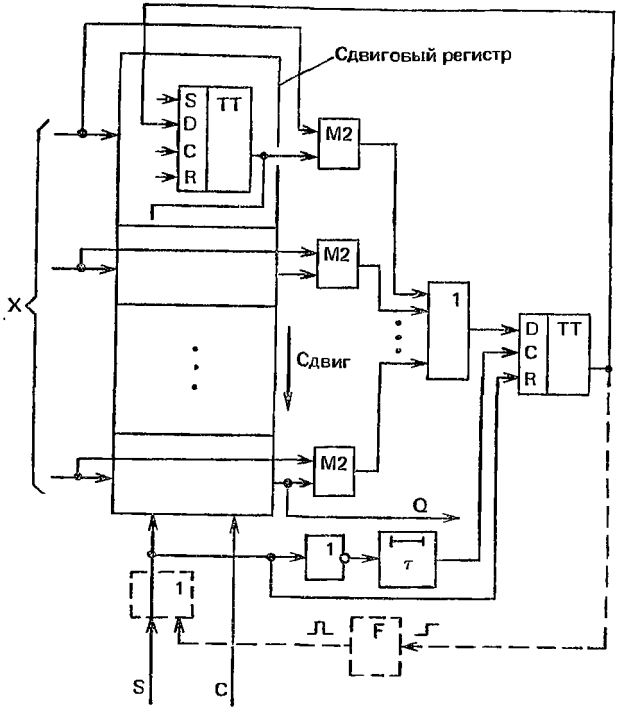
Если от источника информации на линии X поступает «случайная», но относительно медленно меняющаяся информация, то при несовпадении двух последовательных результатов считывания придется провести один или несколько дополнительных сеансов связи до тех пор, пока не будут получены последовательные совпадающие результаты. Если же скорость изменения информации относительно высока или, что то же самое, если процедуры чтения выполняются слишком медленно, то у микроЭВМ не остается никаких шансов отличить истинную информацию от ложной.

Заметим, что почти все имеющиеся неопределенности связаны с разномасштабностью периодов между двумя последовательными считываниями информации (1—5 мс) и периодов нестабильности кодов на линиях X и в регистре (10—50нс). Гораздо более эффективно схемное решение задачи — оно позволяет работать со значительно более «быстрыми» объектами и экономить время за счет почти полного или полного исключения повторного чтения информации вычислительной машиной.

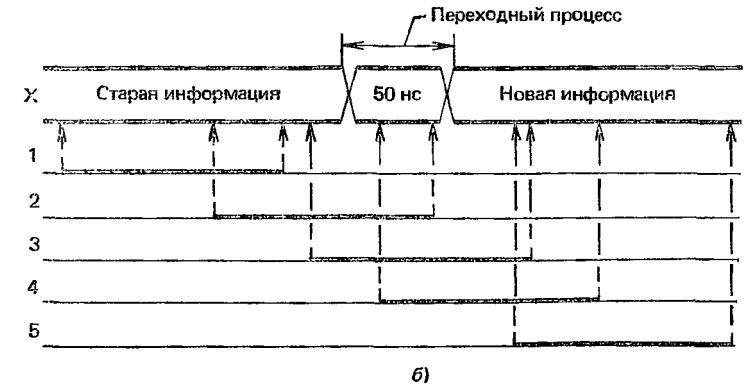
Схемная реализация, первый вариант. На рис. 5.6, *а* показан фрагмент схемы, который следует «подставить» на рис. 5.3. Стыковка этого фрагмента с основной схемой производится в точках S, C, Q, X; при этом из основной схемы исключается «старый» сдвиговый регистр, который заменяется на «новый», построенный по схеме, приведенной на рис. 5.4, но с использованием D-входа триггера верхнего разряда. Не обращая внимания на элементы и связи, показанные штриховыми линиями, отметим, что изменения связаны с введением в схему дополнительной аппаратуры: группы элементов Исключающее ИЛИ (М2), элемента ИЛИ, D-триггера (триггера ошибки), инвертора и элемента задержки.

После введения указанных доработок схема может сама оценивать достоверность информации, получаемой с линий X в начале сеанса связи. Принцип действия тот же, что и ранее: производится двукратный опрос со сравнением результатов, однако интервал времени между опросами на этот раз достаточно мал и лишь незначительно превышает длительность периода неопределенности сигналов на линиях X. Длительность периода между двумя последовательными опросами может быть выбрана, например, равной 100 нс при ожидаемом максимальном периоде нестабильности информации, равном 50 нс.

На рис. 5.6, *б* показаны пять возможных ситуаций (1—5), соответствующих различной последовательности событий. Утолщенными линиями показаны интервалы времени между двумя последовательными считываниями информации. Начало и конец каждой



а)



б)

Рис. 5.6. «Детектор лжи» для выявления недостоверной информации, поступающей от источника по линиям X:
а — схемная реализация; *б* — возможные режимы работы

такой линии соответствуют первому и второму считыванию. Первая ситуация возникает при двух последовательных считываниях одной и той же «старой» информации, которая воспринимается как истинная. Вторая, третья и четвертая ситуации характеризуются несовпадением считанной информации и расцениваются как ошибочные, хотя третья ситуация в действительности дает два достоверных результата считывания. При выявлении ошибочной ситуации процесс двойного считывания повторяется с самого начала по инициативе микроЭВМ. Пятая ситуация соответствует устойчивому считыванию «новой» информации и расценивается как безошибочная.

В исходном состоянии (рис. 5.6, *а*) S=1, поэтому триггер ошибки поддерживается в состоянии лог. 0. Сдвиговый регистр находится в режиме слежения за сигналами на линиях X. Эти сигналы беспрепятственно проходят на выходы регистра. В момент начала сеанса связи сигнал S переходит из 1 в 0, регистр «защелкивается» (возможно, приняв неопределенную информацию из-за переходных процессов в линиях X), а сигнал сброса перестает воздействовать на R-вход триггера ошибки.

Информация, только что записанная в регистр, сравнивается с «новейшей» информацией на входах X. Результат сравнения поступает на D-вход триггера ошибки и переписывается в него по фронту сигнала с выхода элемента задержки. Этот сигнал формируется через 180 нс после поступления лог. 0 на вход S : 100 нс — время между считываниями, 60 нс — суммарное время «успокоения» регистра, элементов Исключающее ИЛИ и элемента ИЛИ, а 20 нс — запас, достаточный для надежного восприятия результата сравнения триггером ошибки.

Если результаты совпали, то на D-вход триггера ошибки поступает сигнал лог. 0 и состояние триггера не меняется. Если «новейшая» информация отличается от той, которая была зафиксирована в регистре на 100 нс ранее, то к моменту поступления фронта сигнала на С-вход триггера ошибки на его D-входе уже в течение, как минимум, 20 нс присутствует лог. 1, поэтому триггер ошибки устанавливается в состояние лог. 1 и сохраняет это состояние на протяжении всего сеанса связи.

МикроЭВМ, как было показано, производит серию сдвигов и в качестве последнего, дополнительного бита принимает бит, хранящийся в триггере ошибки; этот триггер по существу является дополнительным разрядом сдвигового регистра. Получив ту или иную информацию с триггера ошибки, микроЭВМ узнаёт о том, можно ли верить считанному с линий X коду. При наличии признака ошибки микроЭВМ заканчивает текущий сеанс связи и проводит еще один сеанс. Сигнал S вновь переходит из состояния лог. 1 в состояние лог. 0, регистр фиксирует код и т. д. Повторные сеансы в данном случае сравнительно редки и необходимы только в конфликтных ситуациях.

Отметим, что в схеме рис. 5.6, *а* информация из триггера ошибкв поступает в микроЭВМ вслед за полезной информацией, и если триггер ошибки установлен в 1, то получается, что переданное сообщение ложно и его напрасно передавали. Поэтому было бы целесообразно доработать эту схему с тем, чтобы сначала передавать бит из триггера ошибки, а затем, если ошибки нет, полезную информацию.

Схемная реализация, второй вариант. Введение доработок, пока-

занных на рис. 5.6, *а* штриховыми линиями (введены одновибратор и двухвходовый элемент ИЛИ), позволяет в конфликтных ситуациях не прибегать к помощи микроЭВМ и производить серию повторных опросов до тех пор, пока ие будет зафиксирован достоверный результат.

В исходном состоянии S=1, на выходе одновибратора присутствует сигнал лог. 0. Если в начале сеанса связи триггер ошибки установился в состояние лог. 1 (как было описано ранее), то на выходе одновибратора формируется положительный импульс, который проходит через двухвходовый элемент ИЛИ и имитирует конец текущего сеанса связи (S=1) и начало повторного сеанса (S=0). Триггер ошибки при этом устаиавливается в состояние лог. 0, а сдвиговый регистр принимает новую информацию. Далее процесс повторяется, и если справедливы временны́е соотношения, приведенные на рис. 5.6, *б*, а длительность импульса на выходе одновибратора составляет, иапример, 150 нс, то при повторной регистрации будет зафиксироваи заведомо устойчивый результат.

Если по каким-либо причинам триггер ошибки вновь зарегистрирует несовпадение результатов опроса, то поиск будет продолжен до получения достоверного результата. В зависимости от максимально допустимого времени поиска достоверной информации микроЭВМ «теоретически» должна выдерживать бо́льшую или меньшую паузу между моментом t_0 запуска схемы и моментом t_3 опроса первого бита (см. рис. 5.5). Однако практически из-за разномасштабности событий вариациями времени поиска можно пренебречь. Другими словами, микроЭВМ в силу собственной инерционности и из-за сравнительно низкого быстродействия оптроииых схем может «не подозревать» о существовании быстрого схемиого механизма регистрации достоверных данных.

Третий вариант [210] схемного решения (рис. 5.7). В отличие от второго варианта, он обеспечивает более высокую скорость пере-

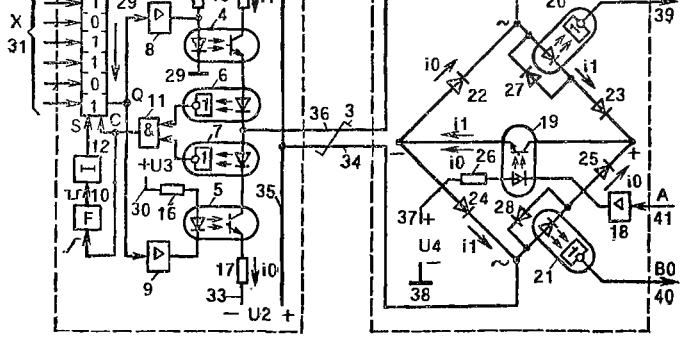


Рис. 5.7. Третий вариант схемного решения устройства для связи микроЭВМ с удаленным источником информации

дачи данных за счет исключения неудачных попыток возбуждения тока в линии связи. При этом, однако, источники питания линии связи физически приближены к источнику информации, что не всегда удобно при практической реализации схемы.

Устройство для сопряжения источника информации с микроЭВМ содержит блок 1 связи с источником информации и блок 2 связи с микроЭВМ, соединенные двухпроводной линией связи 3.

Блок 1 содержит передающие оптроны 4, 5, приемные оптроны 6, 7, усилители 8, 9, формирователь импульса 10, элемент И 11, элемент 12 задержки, сдвиговый регистр 13 и токозадающие резисторы 14 — 17.

Блок 2 содержит усилитель 18, передающий оптрон 19, приемные оптроны 20, 21, диоды 22—25, включенные по схеме двухполупериодного выпрямителя, и токозадающий резистор 26. Защитные диоды 27 и 28 вводятся в схему в тех случаях, когда обратное напряжение, приложенное к входным цепям оптронов 20 и 21, оказывается выше предельно допустимого значения. Усилители 8, 9 и 18 выполнены по схеме с открытым коллектором.

Сдвиговый регистр 13 работает в двух режимах. При наличии сигнала лог. 1 на режимном входе S регистр подготовлен к приему информации (кода X) с разрядных входов 31. Прием производится по положительному фронту сигнала С синхронизации, т. е. при переходе этого сигнала из состояния лог. 0 (напряжение низкого уровня) в состояние лог. 1 (напряжение высокого уровня). При наличии сигнала лог. 0 на режимном входе S регистр подготовлен к сдвигу информации на один разряд в направлении, показанном на рис. 5.7 стрелкой. Сдвиг производится по положительным фронтам сигнала С. Освободившиеся разряды регистра заполняются произвольной информацией (0 или 1) в зависимости от конкретного схемного решения.

Формирователь 10 вырабатывает на выходе отрицательный (по логическому значению) импульс при поступлении на его вход положительного фронта сигнала С. Длительность отрицательного импульса на выходе формирователя 10 выбирается большей, чем длительность одного сеанса связи микроЭВМ с источником информации (в расчете на наихудший случай).

Элемент 12 задержки предотвращает «гонки», которые в его отсутствие могли бы возникнуть при совпадающих или близких по времени сигналах С и S на входах сдвигового регистра 13. При поступлении первого (в сеансе связи) положительного фронта сигнала С регистр 13 находится в режиме приема информации (S=1) и лишь через некоторое время, определяемое элементом 12 задержки, регистр переключается в режим сдвига (S=0) по положительным фронтам последующих сигналов С. Элемент 12 задержки может быть выполнен в виде четырех или шести последовательно включенных элементов НЕ той же серии элементов, которая выбрана для реализации регистра 13.

Приемные оптроны 6, 7, 20, 21 вырабатывают на своих информационных выходах сигналы лог. 0 при излучении света светодиодами этих оптронов. Если светодиод выключен, то на информационном выходе оптрона формируется сигнал лог. 1. Транзисторы передающих оптронов 4, 5, 19 находятся в режиме насыщения (с эквивалентным сопротивлением «коллектор — эмиттер» порядка 10 Ом) при одновременном выполнении двух условий: соответствующий данному

транзистору светодиод излучает свет и потенциал коллектора более положителен, чем потенциал эмиттера.

Источники напряжения U1 и U2 предназначены соответственно для задания тока лог. 1 (i1) и тока лог. 0 (i0) в линии связи 3 при передаче информации к микроЭВМ. Эти источники, по существу, представляют собой один источник напряжения со средней точкой 35.

Источник напряжения U3 обеспечивает питание части схемы, непосредственно связанной с источником информации, в том числе, питание выходных элементов НЕ (инверторов) оптронов 6 и 7. Источник напряжения U4 обеспечивает питание части схемы, связанной с микроЭВМ, в том числе питание выходных элементов НЕ (инверторов) оптронов 20 и 21.

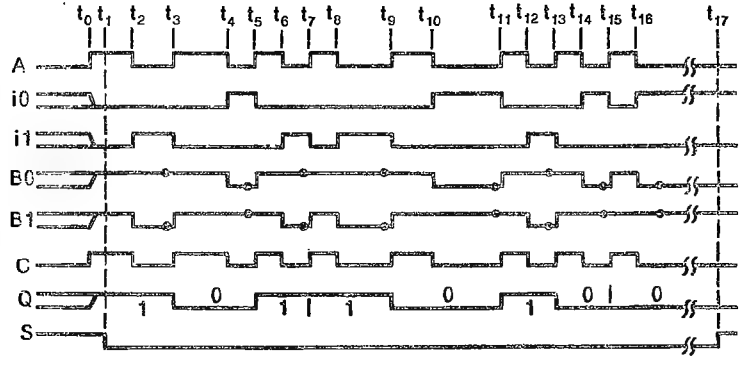


Рис. 5.8. Временные диаграммы передачи информации через устройство (рис. 5.7).

Повышение быстродействия достигнуто за счет исключения неудачных попыток генерации тока в линии связи. Точками показаны моменты опроса

Использование оптронов обеспечивает повышенную помехозащищенность устройства за счет гальванической развязки линии связи 3 с источниками напряжения U3 и U4. Линия связи 3 выполняется витой парой проводов; ее длина может достигать нескольких сотен метров.

Рассмотрим процесс передачи данных от удаленного источника информации к микроЭВМ.

В исходном состоянии (рис. 5.8, период времени до момента t_0) из выходного порта микроЭВМ на вход 41 блока 2 подан сигнал A=0, на выходе усилителя 18 сформировано напряжение низкого уровня, светодиод оптрона 19 излучает свет, транзистор оптрона 19 поддерживается в состоянии насыщения.

На выходе формирователя 10 сформирован сигнал лог. 1 (предыдущий сеанс связи, если он был, окончен), следовательно, регистр 13 подготовлен к приему информации с входов 31. На выходе Q

младшего разряда регистра 13 присутствует сигнал лог. 0 или лог. 1. Если Q=0, то светодиод оптрона 5 включен, а светодиод оптрона 4 выключен. Если Q=1, то светодиод оптрона 4 включен, а светодиод оптрона 5 выключен.

При Q=0 в устройстве протекает ток i0 по цепи источника напряжения U2: шина 35 — провод 34 — входная цепь оптрона 21 — диод 25 — транзистор оптрона 19 — диод 22 — провод 36 — входная цепь оптрона 7 — транзистор оптрона 5 — резистор 17 — шина 33. При этом на выходах оптронов 7 и 21 сформированы сигналы лог. 0, а на выходах оптронов 6 и 20 — сигналы лог. 1. На выходе элемента И 11 сформирован сигнал лог. 0, так как на одном из его входов присутствует сигнал лог. 0.

При Q=1 в устройстве протекает ток i1 по цепи источника напряжения U1: шина 32 — резистор 15 — транзистор оптрона 4 — входнаи цепь оптрона 6 — провод 36 — входная цепь оптрона 20 — диод 23 — транзистор оптрона 19 — диод 24 — провод 34 — шина 35. При этом на выходах оптронов 6 и 20 сформированы сигналы лог. 0, а на выходах оптронов 7 и 21 — сигналы лог. 1. Сигнал синхронизации С регистра 13 поддерживается в состоянии лог. 0.

Сеанс связи микроЭВМ с удаленным источником информации начинается в момент t_0 (рис. 5.8) по инициативе микроЭВМ, которая выдает сигнал А=1 на вход усилителя 18, что вызывает выключение светодиода оптроиа 19 и прекращение тока i0 или i1 в линии связи 3 (линия связи размыкается).

Прекращение тока в линии связи 3 приводит к формированию на обоих входах элемента И 11 сигналов лог. 1 (светодиоды оптронов 6 и 7 выключены), следовательно, на выходе элемента И 11 формируется положительный перепад напряжения. Так как в момент t_0 S=1, регистр 13 находится в режиме приема информации. По фронту сигнала С код X записывается в регистр. В примере, приведенном на рис. 5.7, X=00101101.

В момент t_1 регистр 13 переключается в режим сдвига (S=0) и находится в этом режиме в течение времени, заведомо превышающего длительность сеанса связи микроЭВМ с источником информации (до момента t_{17}).

Длительность первого положительного импульса А (период времени $t_0 — t_2$) должна быть достаточной для полного выключения оптроиов, по которым в исходном состоянии протекал ток, и для завершения переходных процессов в регистре 13. Оптроны обладают относительно большой инерционностью при выключении, поэтому минимальная длительность первого и последующих положительных импульсов А может составлять, например, 20 мкс.

В момент t_2 микроЭВМ формирует на входе 41 устройства сигнал А=0, который, пройдя через усилитель 18, включает светодиод оптрона 19; транзистор этого оптрона переходит в режим насыщения и более не препятствует протеканию тока i1 (в младшем разряде регистра 13 записана лог. 1, светодиод оптрона 4 включен). После окончания переходных процессов ток i1 принимает установившееся значение, на выходе элемента И 11 формируется сигнал лог. 0, сигнал С переходит в состояние лог. 0 (что не оказывает воздействия на регистр 13), на выходе В1 формируется сигнал лог. 0, что воспринимается микроЭВМ как признак наличия лог. 1 в младшем разряде регистра 13.

Выждав время, заведомо превышающее период установления истинных сигналов на выходах 39 и 40 устройства (несколько мик-

росекунд), микроЭВМ считывает сигналы с этих выходов и, обнаружив, что B1=0, а B0=1, «узиаёт», что младший бит кода X имеет единичное значение. Моменты считывания информации с выходов B0 и B1 показаны на временных диаграммах (рис. 5.8) точками.

После считывания первого бита, в момент t_3 микроЭВМ вновь размыкает линию связи 3 сигналом A=1. На входе C регистра 13 формируется положительный фронт сигнала, и, поскольку S=0, в регистре производится сдвиг информации на один разряд «вниз». Предыдущий разряд, ранее считаный микроЭВМ, теряется. (Если «закольцевать» регистр 13, т. е. соединить его выход Q с верхним входом сдвига, то считываемая информация будет вращаться по кольцу и ее можно будет считывать многократно, например, с целью повышения достоверности передачи.) В примере, приведенном на рис. 5.7, после первого сдвига сигнал Q принимает нулевое значение, в результате в блоке 1 создаются условии для протекании тока i0, который, однако, пока не может течь, так как транзистор оптрона 19 в блоке 2 выключен.

В момент t_4 микроЭВМ снимает запрет на протекание тока по линии связи 3 (A=0) и затем, опросив выходы 39 и 40 (B0=0, B1= =1), принимает к сведению, что после первого сдвига в младшем разряде регистра 13 присутствует сигнал лог. 0.

В момент t_5 линия связи 3 размыкается, информация в регистре 13 вновь сдвигается на один разряд и в точке Q формируется сигнал лог. 1, что создает предпосылку для протекания тока i1, когда это будет разрешено сигналом A=0 от микроЭВМ.

В момент t_6 микроЭВМ разрешает протекание тока по линии связи 3 и следит за ответной реакцией на линиях 39 и 40. В данном случае B1=0, B0=1, поэтому в память микроЭВМ заносится единичный бит.

Далее процесс продолжается, и после момента t_{16} микроЭВМ производит последний опрос выходов 39 и 40 (см. последнюю справа пару точек на временных диаграммах сигналов B0 и B1, рис. 5.8). Полученные в результате взаимодействия микроЭВМ с устройством биты программным путем собираются в единое слово, возможно, занимающее несколько ячеек памяти. Этот результат можно рассматривать как «перенос» кода X с параллельных выходов удаленного источника информации в память микроЭВМ, что и требовалось.

В момент t_{17} заканчивается отрицательный импульс на режимном входе S регистра 13 и устройство переходит в описанное ранее исходное состоиние.

Из временных диаграмм, приведенных на рис. 5.8, следует, что при правильной работе устройства сигналы B0 и B1 в моменты их опроса (см. точки на диаграммах) противофазны. Это позволяет оперативно обнаруживать ошибки (типа сбой или отказ) непосредственно в процессе передачи информации через устройство: при наличии одинаковых сигналов B0 и B1 (оба сигнала нулевые или единичные) в момент опроса можно утверждать, что устройство работает неправильно и передаваемая информация недостоверна.

Улучшения второго и третьего вариантов схемных решений. В этих вариантах использовался положительный и отрицательный ток в линии связи. Предлагается использовать также «третье состояние» линии связи (выключенное) для передачи в микроЭВМ признака «досрочного» окончания сеанса связи.

Сеанс связи может быть завершен досрочно, если существует «предварительная договоренность» (заложенная в программу, выпол-

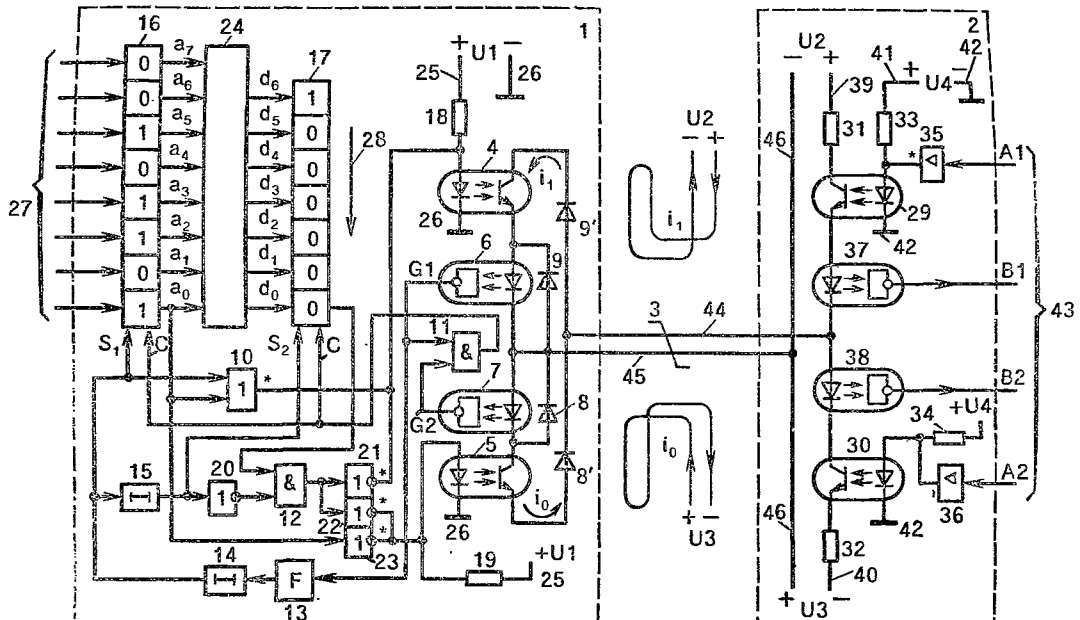


Рис. 5.9. Усовершенствование второго варианта схемного решения (см. рис. 5.3).

Логический блок 24 позволяет выявлять условия досрочного окончания сеанса связи

няемую микроЭВМ) о том, что́ означает «внеплановое» размыкание линии связи источником информации. Например, размыкание линии связи в процессе передачи кодового слова может являться признаком того, что в оставшихся (непереданных) разрядах слова содержатся одинаковые биты (все нули или все единицы), противоположные только что переданному биту. Другой пример: линия размыкается, если оставшиеся биты передаваемого кодового слова совпадают с соответствующими битами ранее переданного кодового слова. Далее будут рассмотрены схемные решения, соответствующие этим двум примерам; однако следует учесть, что число таких примеров практически безгранично: переход линии связи в «третье состояние» может являться требованием прерывания, признаком ошибки, указателем конца передаваемого слова, если оно имеет переменную длину, и т. п.

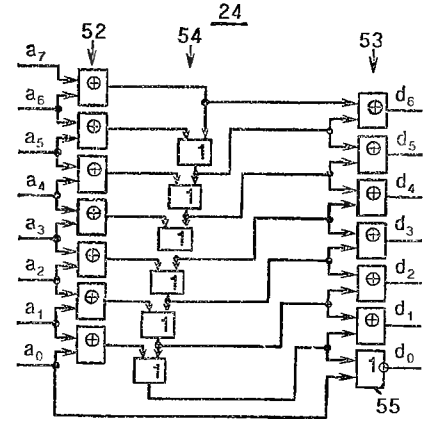


Рис. 5.10. Вариант схемы логического блока 24 (см. рис. 5.9)

Досрочное окончание сеанса связи из-за обнаружения цепочки одинаковых бит (000...0 или 111...1) в конце передаваемого слова [113]. Схема устройства для сопряжения источника информации с микроЭВМ приведена на рис. 5.9. Устройство содержит блок 1 связи с источником информации и блок 2 связи с микроЭВМ, соединенные двухпроводной линией связи 3. Блок 1 содержит передающие оптроны 4, 5, приемные оптроны 6, 7, диоды 8, 8′, 9, 9′, элемент ИЛИ 10, элементы И 11, 12, формирователь импульса 13, элементы задержки 14, 15, сдвиговые регистры 16, 17, токозадающие резисторы 18, 19, элементы НЕ 20—23 и логический блок 24. Направление сдвига информации в регистрах 16 и 17 показано стрелкой 28. Освободившиеся при сдвиге разряды регистров заполняются нулями.

Блок 2 содержит передающие оптроны 29, 30, резисторы 31—34, усилители 35, 36, приемные оптроны 37, 38.

Логический блок 24 (рис. 5.10) содержит группы 52 и 53 элементов Исключающее ИЛИ, группу 54 элементов ИЛИ и элемент ИЛИ — НЕ 55.

Логический блок 24 (рис. 5.9, 5.10) анализирует код $a_0 — a_7$ с целью выявления в нем цепочки одинаковых бит, расположенных в верхней (по схеме) части регистра 16. В примере, приведенном на рис. 5.9, в верхних разрядах регистра 16 записаны два одинаковых бита: $a_7 = a_6 = 0$. Этот факт воспринимается логическим блоком 24 и он формирует на выходе d_6 сигнал лог. 1. В общем случае при $a_n = a_{n-1} = ... = a_1$ в разряд d_1 регистра 17 будет записана 1, а в ос-

тальные разряды — 0. Другими словами, в регистре 17 помечается «начало конца» из одинаковых бит в кодовом слове X, записанном в регистр 16. Исключением является кодовая комбинация $a_n = a_{n-1} = \ldots = a_0 = 1$, в ответ на которую блок 24 формирует нулевой код $d_{n-1} = d_{n-2} = \ldots = d_0 = 0$. Работа логического блока поясняется табл. 5.1.

Знак «×» в табл. 5.1 означает произвольное состояние (0 или 1). Таким образом, в таблице перечислены все возможные кодовые комбинации в регистре 16. Например, комбинация 11000110 описывается четвертой строкой табл. 5.1, комбинация 11101100 — шестой строкой и т. д.

Группа 52 логических элементов Исключающее ИЛИ (рис. 5.10) выделяет во входном коде несовпадающие позиции. Группа элементов ИЛИ 54 размножает «верхнюю» единицу с выходов элементов 52, а выходные элементы 53 и 55 определяют границу между полученными группами нулей и единиц, т. е. производят окончательную обработку сигналов согласно табл. 5.1.

Логический блок 24 может быть выполнен в виде ПЗУ, в дан-

Таблица 5.1

a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	d_6	d_5	d_4	d_3	d_2	d_1	d_0
0	1	×	×	×	×	×	×	0	0	0	0	0	0	0
1	0	×	×	×	×	×	×	0	0	0	0	0	0	0
0	0	1	×	×	×	×	×	1	0	0	0	0	0	0
1	1	0	×	×	×	×	×	1	0	0	0	0	0	0
0	0	0	1	×	×	×	×	0	1	0	0	0	0	0
1	1	1	0	×	×	×	×	0	1	0	0	0	0	0
0	0	0	0	1	×	×	×	0	0	1	0	0	0	0
1	1	1	1	0	×	×	×	0	0	1	0	0	0	0
0	0	0	0	0	1	×	×	0	0	0	1	0	0	0
1	1	1	1	1	0	×	×	0	0	0	1	0	0	0
0	0	0	0	0	0	1	×	0	0	0	0	1	0	0
1	1	1	1	1	1	0	×	0	0	0	0	1	0	0
0	0	0	0	0	0	0	1	0	0	0	0	0	1	0
1	1	1	1	1	1	1	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
1	1	1	1	1	1	1	1	0	0	0	0	0	0	0

ном примере — емкостью 256 7-разрядных ячеек. Сигналы $a_0 - a_7$ в этом случае поступают на адресные входы ПЗУ, а сигналы $d_0 - d_6$ формируются на его выходах данных. Можно использовать ПЛМ (см. § 1.4); при этом, как следует из табл. 5.1, в матрице элементов И — ИЛИ будет занято 13 логических произведений, по числу единиц в правой части таблицы. В общих словах работа устройства заключается в следующем.

В процессе передачи информации блок 1 производит последовательный сдвиг кодов, первоначально записанных в регистры 16 и 17. Если в нижнем разряде регистра 16 в некоторый момент времени присутствует лог. 1, то блок 1 разрешает прохождение через

него тока i_1. В противном случае блок подготовлен к прохождению через него тока i_0, а для тока i_1 цепь разомкнута. МикроЭВМ при передаче каждого бита тестирует линию связи 3, пытаясь возбудить ток i_1 или i_0. Если удалось вызвать ток i_1 (i_0), то это означает, что передается лог. 1 (лог. 0). Если не удалось вызвать ток i_1 (i_0), то микроЭВМ пытается вызвать ток противоположного направления i_0 (i_1). Эта повторная попытка может оказаться успешной, и микроЭВМ регистрирует соответствующий бит, а информация в регистрах 16 и 17 по окончании импульса тока сдвигается на один разряд вниз. Если повторная попытка также оказалась безуспешной, то микроЭВМ «узнаёт», что блок 1 неспособен транслировать через себя ток ни в том, ни в другом направлении из-за того, что в нижнем разряде регистра 17 присутствует лог. 1. А это, в свою очередь, означает, что оставшиеся биты передаваемого кода имеют одинаковые значения, противоположные только что принятому биту. Поэтому нет смысла тратить время на передачу этих одинаковых бит, сеанс связи досрочно заканчивается, а микроЭВМ доопределяет недостающие биты в принятом коде нулями или единицами, в зависимости от предыстории. Таким образом, среднее время передачи кода от источника информации к микроЭВМ сокращается. Далее приведено детальное описание работы устройства.

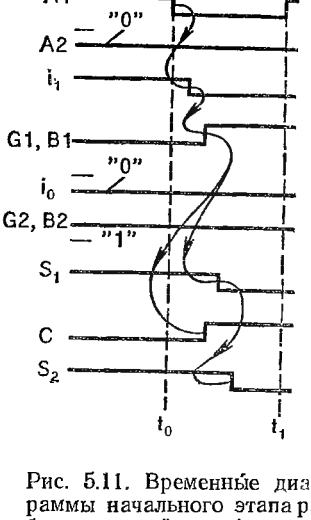


Рис. 5.11. Временные диаграммы начального этапа работы устройства (см. рис. 5.9)

В исходном состоянии (рис. 5.11, период времени до момента t_0) из выходного порта микроЭВМ на входы блока 2 поданы сигналы A1=1 и A2=0, поэтому светодиод оптрона 29 включен, а светодиод оптрона 30 выключен, так что потенциал провода 44 более положителен, чем потенциал провода 45.

В блоке 1 в исходном состоянии $S_1=S_2=1$ (предыдущий сеанс связи, если он был, завершен, отрицательный импульс на выходе формирователя 13 закончился), регистр 16 транслирует сигналы с входов 27 на свои выходы, на выходе элемента НЕ 20 поддерживается сигнал лог. 0, который запирает элемент И 12 и выключает элементы НЕ 21 и 22 с открытым коллектором (выходы элементов с открытым коллектором помечены на рис. 5.9 знаками *). Сигнал $S_1=1$ воздействует на элемент ИЛИ 10, так что выходной транзистор этого элемента выключен. Поэтому светодиод оптрона 4 включен, транзистор этого оптрона насыщен и созданы благоприятные условия для протекания тока i_1 по цепи источника напряжения U2: шина 39 — резистор 31 — насыщенный транзистор оптрона 29 — светодиод оптрона 37 — провод 44 линии связи 3 — насыщенный транзистор оптрона 4 — светодиод оптрона 6 — провод 45 линии связи

3 — шина 46. Этот путь протекания тока i_1 условно показан на рис. 5.9 над линией связи 3. Ток через транзистор оптрона 5 практически равен нулю.

Сеанс связи начинается в момент t_0 (рис. 5.11) по инициативе микроЭВМ, которая выдает сигнал A1=0 на вход усилителя 35 блока 2, что вызывает прекращение тока i_1 в линии связи 3 из-за выключения транзистора оптрона 29. На выходах оптронов 6 и 37 формируются единичные сигналы G1=1 и B1=1. При этом срабатывает элемент И 11 и формируется положительный фронт сигнала C, который, однако, не воздействует на состояние регистров 16 и 17, так как на их входах S_1 и S_2 все еще присутствуют сигналы лог. 1. Формирователь импульса 13 по сигналу G1=1 вырабатывает «широкий» отрицательный импульс, который через элемент задержки 14 передается на вход S_1 регистра 16 и далее через элемент задержки 15 — на вход S_2 регистра 17.

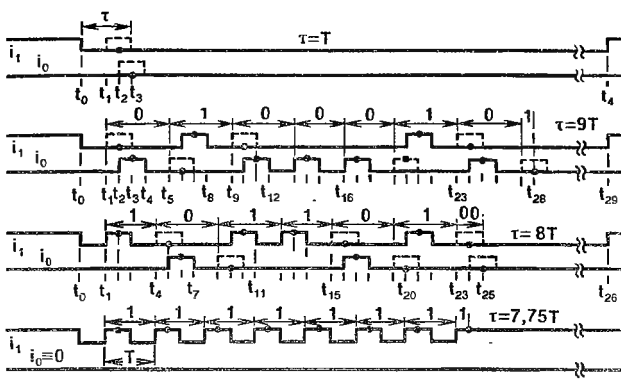


Рис. 5.12. Временные диаграммы работы устройства (см. рис. 5.9) при передаче различных кодов

Таким образом, в результате выдачи сигнала A1=0 входная информация с группы входов 27 запомнилась в регистре 16, а результат работы логического блока 24 зафиксирован в регистре 17. Ток в линии связи 3 отсутствует, инициатива по управлению устройством вновь переходит к микроЭВМ.

После паузы, гарантирующей окончание переходных процессов в устройстве, в момент t_1 (рис. 5.11) микроЭВМ пытается вновь вызвать протекание в линии 3 тока i_1 сигналом A1=1. Однако исход этой попытки зависит от информации, зафиксированной в регистрах 16 и 17. Далее рассмотрена передача четырех характерных кодов.

1. Передача кода 000...0 (рис. 5.12, *а*). В момент t_1 микроЭВМ пытается вызвать в линии связи 3 ток i_1 включением транзистора оптрона 29 (A1=1), однако эта попытка оказывается безуспешной («желаемый» импульс тока показан на временной диаграмме штриховой линией). Действительно, в начале сеанса связи, в период времени, близкий t_0, в ответ на нулевой код в регистре 16 логический блок 24 сформировал, а регистр 17 зафиксировал код, содержащий

лог. 1 в разряде d_0 и лог. 0 в остальных разрядах (см. предпоследнюю строку табл. 5.1). Сигнал лог. 1 из нижнего разряда регистра 17 проходит через элемент И 12 и поддерживает элементы НЕ 21 и 22 во включенном состоянии, так что светодиоды оптронов 4 и 5 не излучают свет. Блок 1, таким образом, полностью лишен возможности пропускать ток в том или ином направлении по линии связи 3, так как передающие оптроны 4 и 5 выключены.

В момент t_2 после паузы, достаточной для завершения возможных переходных процессов, микроЭВМ опрашивает состояние линии В1 и «узнаёт», что тока i_1 нет (В1=1). (Моменты опроса линии связи 3 машиной показаны на рис. 5.12 точками.) Теперь микроЭВМ пытается вызвать ток i_0, выдавая на входы устройства сигналы А1=0, А2=1. Ток i_0 (см. импульс, показанный штриховой линией) при благоприятных условиях мог бы протекать по цепи: шина 46 — провод 45 линии связи 3 — светодиод оптрона 7 — насыщенный транзистор оптрона 5 (в данном случае этот транзистор выключен) — провод 44 линии связи 3 — светодиод оптрона 38 — насыщенный транзистор оптрона 30 — резистор 32 — шина 40 (см. схему распространения тока i_0, приведенную на рис. 5.9 под линией связи 3).

В момент t_3 микроЭВМ проверяет наличие тока i_0, опрашивая линию В2, и убеждается в том, что ток i_0 также не удалось пропустить по линии связи 3 (В2=1).

Факт невозможности пропускания токов i_1 и i_0 через блок 1 в данном случае расценивается машиной как сообщение о том, что во всех разрядах кода на входах 27 присутствуют лог. 0. Поэтому сеанс связи заканчивается досрочно, в момент t_3 и микроЭВМ формирует сигналы А1=1 и А2=0, соответствующие исходному состоянию устройства. В момент t_4 заканчивается импульс на выходе формирователя 13 и устройство возвращается в исходное состояние, при котором есть ток i_1 и нет тока i_0.

2. Передача кода, не содержащего одинаковых битов в последних разрядах (рис. 5.12, *б*). В момент t_1 микроЭВМ пытается вызвать ток i_1, но эта попытка оказалась безуспешной, так как $a_0=0$, $S_1=0$, на выходе элемента ИЛИ 10 сформировано напряжение низкого уровня и светодиод оптрона 4 выключен. В момент времени t_2 микроЭВМ узнаёт об отсутствии тока i_1 и пытается вызвать ток i_0. На этот раз попытка оказалась успешной, так как светодиод оптрона 5 излучает свет благодаря выключенному состоянию элементов НЕ 22 и 23. В момент времени t_3 микроЭВМ принимает к сведению наличие тока i_0 (В2=0, следовательно, передавался лог. 0) и далее выключает оптрон 30, в результате ток i_0 прекращается. В блоке 1 срабатывает элемент И 11 и по фронту сигнала С=1 содержимое регистров 16 и 17 сдвигается на один разряд вниз.

В момент t_5 микроЭВМ пытается возбудить ток i_0, но этого не удается сделать, так что в момент t_6 микроЭВМ пытается возбудить ток i_1 — на этот раз попытка оказалась успешной.

Далее процессы протекают аналогично: микроЭВМ тестирует линию связи 3, «отталкиваясь» от значения только что полученного бита, и в случае неудачи ищет противоположный бит. В данном примере код не содержит одинаковых бит в последних разрядах, и поэтому в регистре 17 зафиксирован нулевой код (см. вторую строку табл. 5.1). Поэтому не возникает ситуации, когда выключены светодиоды обоих оптронов 4 и 5, и, следовательно, микроЭВМ обязательно находит условия протекания тока в том или ином направлении и принимает к сведению соответствующий бит.

В момент t_{28} микроЭВМ получает информацию о том, что последний бит не равен нулю — ток i_0 вызвать не удалось. Поэтому микроЭВМ делает вывод о том, что последний бит равен 1; в этом случае уже нет необходимости возбуждать ток i_1. Далее микроЭВМ формирует сигналы A1=1 и A2=0 и в момент t_{29} по окончании импульса на выходе формирователя 13 устройство переходит в исходное состояние. Время передачи кода в примере, представленном на рис. 5.12, *б*, составляет 9 Т, где Т — период следования импульсов тока при удачных сравнениях.

3. Передача кода, содержащего одинаковые биты в последних разрядах (рис. 5.12, *в*). В качестве примера рассматривается код, показанный на рис. 5.9, в котором $a_6=a_7=0$. В этом случае (см. табл. 5.1, третью строку сверху) $d_6=1$, $d_0=d_1=...=d_5=0$. Передача первых шести бит производится так же, как и в предыдущем примере (см. п. 2), методом проб и ошибок. При этом единица последовательно перемещается в регистре 17 вниз и в момент t_{22} она поступает в нижний разряд этого регистра, выключая оптроны 4 и 5.

В момент t_{24} микроЭВМ убеждается в отсутствии тока i_1, а в момент времени t_{25} — в отсутствии тока i_0. Эта ситуация расценивается машиной как сообщение о досрочном окончании сеанса связи — микроЭВМ узнаёт о том, что после только что принятой лог. 1 (a_5) следуют одинаковые биты, противоположные по логическому значению ($a_6=a_7=0$). В момент времени t_{26} устройство переходит в исходное состояние. Время передачи кода равно 8Т.

4. Передача кода 111...1 (рис. 5.12, *г*). При передаче этого кода все попытки вызвать ток i_1 оказываются успешными. Время передачи кода составляет 7,75 Т.

Приведенные четыре примера исчерпывают все возможные ситуации, которые могут возникнуть при передаче кодов от источника информации к микроЭВМ.

Блок-схема алгоритма работы микроЭВМ при проведении сеанса связи с источником информации показана на рис. 5.13.

Блок 74 соответствует исходному состоянию устройства (A1=1, A2=0, B1=0, B2=1), блок 75 — выключению источника тока i_1 и программной задержке, большей или равной 0,5 Т. Далее (блок 76) переменным k и j присваиваются исходные значения, и после этого начинается выполнение циклической части алгоритма. Блок 77 увеличивает на единицу переменную k и присваивает нулевое значение переменной M. Условие M=0 соответствует первому тестированию бита, условие M=1 означает, что уже была одна неудачная попытка вызвать ток (i_1 или i_0) в линии связи 3.

Блок 78 производит включение источника тока i_1 или i_0, блок 79 реализует временну́ю задержку, большую или равную 0,25 Т, для компенсации переходных процессов в устройстве. Блок 80 проверяет наличие тока i_1 или i_0, в зависимости от значения переменной j. Если ток есть, то микроЭВМ присваивает переменной a_k значение j (блок 81), т. е. микроЭВМ принимает очередной бит и далее (блок 82) проверяет, все ли биты приняты (n — разрядность кода на входах 27 устройства). Если все биты приняты, то микроЭВМ переходит к исходному состоянию (блок 89), в противном случае она выключает источник тока, который был включен (блок 83), и после программной задержки, большей или равной 0,75 Т (блок 84), возвращается к выполнению действий, предписанных блоком 77.

Если при проверке наличия тока (блок 80) оказалось, что тока нет, то проверяется условие k=n — 1. Если k=n — 1, то это означает,

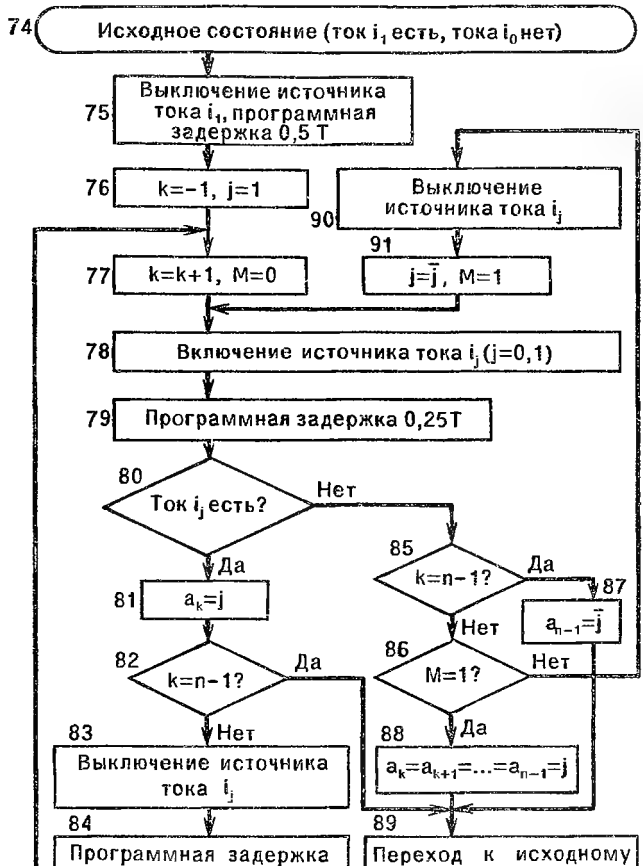


Рис. 5.13. Блок-схема алгоритма работы микроЭВМ при проведении сеанса связи с источником информации (см. рис. 5.9)

что принимается последний бит и, следовательно, его значение соответствует $\bar{j}$. Это значение присваивается разряду a_{n-1} (блок 87), и далее микроЭВМ переходит к исходному состоянию (A1=1, A2==0, блок 89). Если $k \neq n-1$, то в случае повторной проверки (M=1, блок 86) оставшимся битам присваиваются одинаковые значения, равные j (блок 88), и далее производится переход к исходному состоянию. Если M=0, то выключается источник тока, которым не удалось воспользоваться (блок 90), производится подготовка к включению «противоположного» источника тока и отмечается факт

неудачного тестирования бита (блок 91), затем производится переход к блоку 78.

Приведенная на рис. 5.13 блок-схема реализуется в виде программы, хранимой в памяти микроЭВМ и выполняемой при необходимости проведения сеанса связи с источником информации.

Число связей блока 2 с микроЭВМ может быть уменьшено при использовании схемы, показанной на рис. 5.14. В этой схеме линии B1 и B2 объединяются элементом И 56. Такое схемное решение воз-

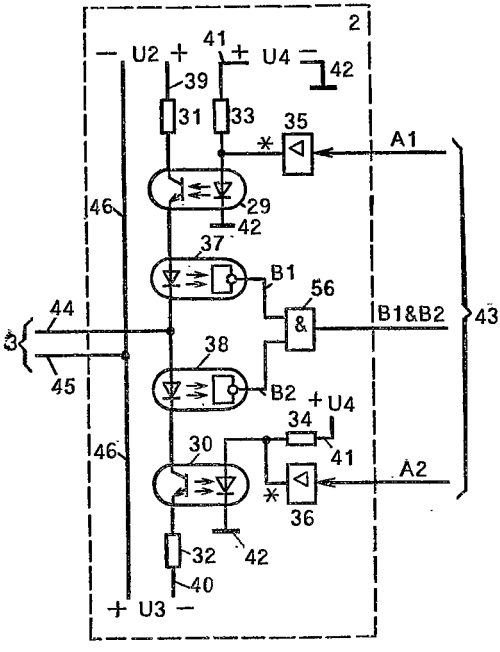


Рис. 5.14. Вариант выполнения блока связи с микроЭВМ (см. рис. 5.9)

Вместо двух выходов B1 и B2 использован один обобщенный выход B1 & B2

можно благодаря тому, что сигналы лог. 0 на этих линиях никогда не формируются одновременно и их можно считывать в нужной последовательности с обобщенного выхода B1&B2.

В табл. 5.2 приведены сравнительные временные характеристики устройств, показанных на рис. 5.3 и 5.9 (без логического блока и с логическим блоком), при передаче 4-разрядных кодов. Из табл. 5.2 следует, что при передаче девяти кодов имеется выигрыш во времени, при передаче трех кодов — проигрыш, а передача оставшихся четырех кодов тем и другим устройством требует одинаковых затрат времени. При равновероятном появлении любых входных ко-

Таблица 5.2

Передаваемый код a_0 a_1 a_2 a_3	Время передачи кода устройством без логического блока, см. рис. 5.3 (число периодов T)	Время передачи кода устройством с логическим блоком, см. рис. 5.9 (число периодов T)	Выигрыш (число периодов T)
0 0 0 0	4,5	1,0	+3,5
0 0 0 1	4,5	4,0	+0,5
0 0 1 0	4,25	4,25	0
0 0 1 1	4,25	3,25	+1
0 1 0 0	4,25	3,5	+0,75
0 1 0 1	4,25	4,5	−0,25
0 1 1 0	4,0	4,25	−0,25
0 1 1 1	4,0	2,25	+1,75
1 0 0 0	4,25	2,0	+2,25
1 0 0 1	4,25	4,0	+0,25
1 0 1 0	4,0	4,25	−0,25
1 0 1 1	4,0	3,25	+0,75
1 1 0 0	4,0	3,0	+1,0
1 1 0 1	4,0	4,0	0
1 1 1 0	3,75	3,75	0
1 1 1 1	3,75	3,75	0
Усредненные результаты	4,125 T	3,4375 T	+0,6875 T

довых комбинаций обеспечивается снижение среднего времени передачи кода (см. последнюю строку табл. 5.2).

Эта же усредненная информация представлена на рис. 5.15 в точке, соответствующей $n=4$ (n — разрядность передаваемых кодов, единица измерения величин $\tau(n)$ и $\tau^*(n)$ — число периодов T сигнала в линии связи). Среднее время передачи кода устройством, показанным на рис. 5.3 (прямая 92), линейно растет с увеличением n. Среднее время передачи кода устройством, показанным на рис. 5.9 (кривая 93), также растет с увеличением n, однако этот рост при $n<6$ (строго говоря, при любом n) нелинеен. В результате с увеличением n накапливается выигрыш, при $n\geqslant 6$ равный одному периоду T сигнала в линии связи. Относительный выигрыш во времени поэтому имеет максимум (кривая 94). При $n=3$ и $n=4$ выигрыш максимален и составляет 16,6 %.

Досрочное окончание сеанса связи из-за обнаружения частичного или полного совпадения новой информации со старой, переданной в предыдущем сеансе связи. При опросе состояния многих объектов, например цифровых датчиков температуры массивного тела, поступающая в микроЭВМ информация меняется медленно, во всяком случае, в старших разрядах. Предлагается использовать такую «инерционность» входной информации для уменьшения числа передаваемых в микроЭВМ бит [16].

Если новый код полностью совпадает с переданным в предыдущем сеансе, то блок связи с источником информации «отказывается» от приглашения к очередному сеансу связи, размыкая линию. Мик-

роЭВМ, в свою очередь, установив факт размыкания линии, считает, что входная информация осталась прежней.

Если новый код отличается от старого только в нескольких младших разрядах, то эти разряды передаются в микроЭВМ обычным путем. Группа старших разрядов, оставшихся неизменными, не передается, так как блок связи с источником информации размыкает линию, а микроЭВМ, обнаружив размыкание, доопределяет значения отсутствующих разрядов ранее принятыми значениями.

Сеансы связи, сокращенные в большей или меньшей степени, могут следовать друг за другом. В частности, блок связи с источником информации может постоянно «отказываться» от взаимодействия с микроЭВМ из-за постоянства входного кода При этом, однако, в силу разных причин микроЭВМ может «забыть» предысторию (например, если напряжение питания мнкроЭВМ временно отключалось) и не знать, чем доопределять отсутствующее слово или его часть. Иным словами, может потеряться взаимопонимание между микроЭВМ и блоком связи с источником информации.

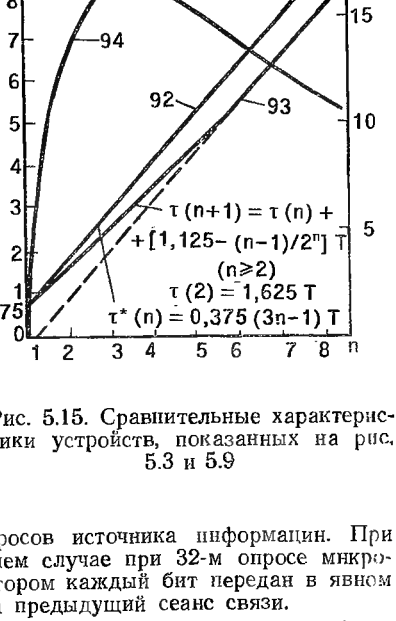


Рис. 5.15. Сравнительные характеристики устройств, показанных на рис. 5.3 и 5.9

Для исключения таких потерь предлагается периодически, например в каждом 32-м сеансе связи, передавать полное слово, даже если оно в точности совпадает с ранее переданным. Это позволяет установить «информационный синхронизм» между микроЭВМ и блоком связи с источником информации. После включения напряжения питания или в других подобных ситуациях микроЭВМ проводит серию опросов источника информации. При этом гарантируется, что в худшем случае при 32-м опросе мнкроЭВМ получит полный код, в котором каждый бит передан в явном виде (0 или 1), без «ссылок» на предыдущий сеанс связи.

Схема, показанная на рис. 5.16, получена в результате доработки схемы, показанной на рис. 5.7. Доработки связаны с введением дополнительной аппаратуры сравнения «старого» и «нового» кода для выявления возможности досрочного окончания сеанса связи. (Примерно так же может быть доработана схема, приведенная на рис. 5.3.)

Принцип действия схем (рис. 5.16 и 5.7) при передаче полных кодов (без сокращений) одинаков, поэтому далее подробно не рассматривается. Отметим лишь те особенности схемы, которые существенны для понимания принципа работы дополнительной аппаратуры.

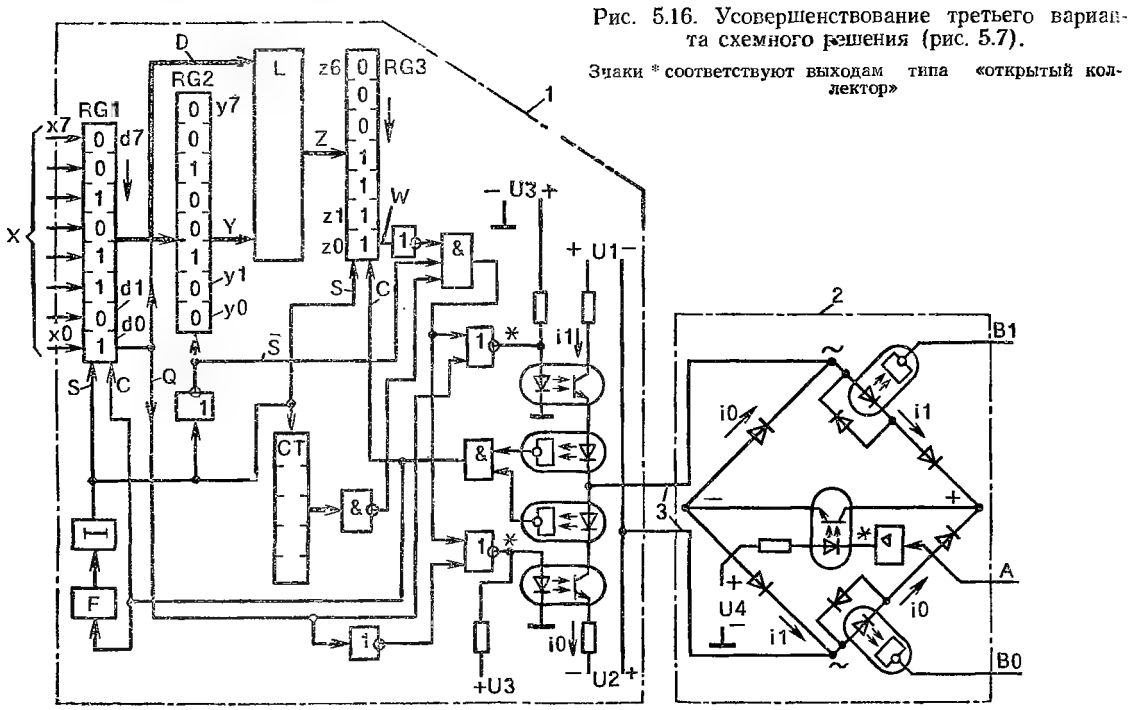


Рис. 5.16. Усовершенствование третьего варианта схемного решения (рис. 5.7).

Знаки * соответствуют выходам типа «открытый коллектор»

Сдвиговый регистр RG1, как и в схеме, показанной на рис. 5.7, работает в двух режимах. При S=1 по положительному фронту сигнала C производится прием кода X в регистр. При S=0 по каждому положительному фронту сигнала C производится сдвиг информации на один разряд вниз. Освободившиеся разряды регистра RG1 заполняются произвольной информацией.

Параллельный регистр RG2 принимает информацию по положительному фронту сигнала $\overline{S}$ и хранит ее до момента поступления следующего положительного фронта этого сигнала. Сдвиговый регистр RG3 выполнен иначе, чем регистр RG1; его структура соответствует схеме, приведенной на рис. 5.4, за исключением того, что

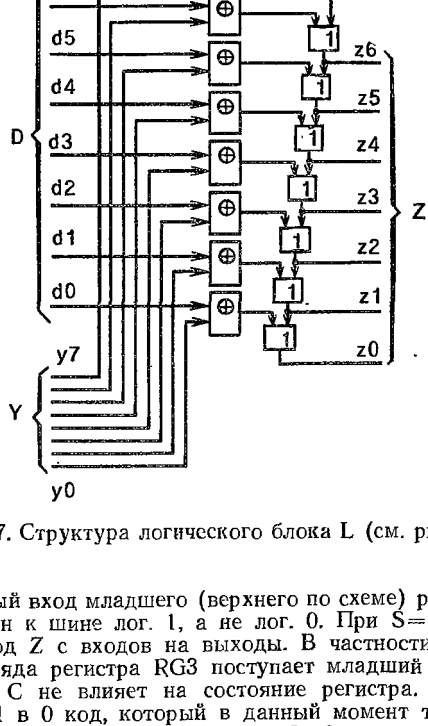


Рис. 5.17. Структура логического блока L (см. рис. 5.16)

информационный вход младшего (верхнего по схеме) разряда должен быть подключен к шине лог. 1, а не лог. 0. При S=1 регистр RG3 транслирует код Z с входов на выходы. В частности, на выход W младшего разряда регистра RG3 поступает младший разряд z0 кода Z. Сигнал C не влияет на состояние регистра. При переходе сигнала S из 1 в 0 код, который в данный момент транслировался через регистр, запоминается в нем. При S=0 регистр RG3 способен сдвигать информацию вниз по положительным фронтам сигнала C. Существенно, что освободившиеся разряды регистра RG3 заполняются единицами (в отличие от схемы, показанной на рис. 5.4). Это предотвращает потерю старшего разряда при передаче полного кода.

Логический блок L (рис. 5.17) выполняет функции сравнения кодов в соответствии с табл. 5.3.

Таблица 5.3

Соотношения бит в кодах D и Y								Выходной код Z						
d7	d6	d5	d4	d3	d2	d1	d0	z6	z5	z4	z3	z2	z1	z0
$\overline{y7}$	×	×	×	×	×	×	×	1	1	1	1	1	1	1
y7	$\overline{y6}$	×	×	×	×	×	×	1	1	1	1	1	1	1
y7	y6	$\overline{y5}$	×	×	×	×	×	0	1	1	1	1	1	1
y7	y6	y5	$\overline{y4}$	×	×	×	×	0	0	1	1	1	1	1
y7	y6	y5	y4	$\overline{y3}$	×	×	×	0	0	0	1	1	1	1*
y7	y6	y5	y4	y3	$\overline{y2}$	×	×	0	0	0	0	1	1	1
y7	y6	y5	y4	y3	y2	$\overline{y1}$	×	0	0	0	0	0	1	1
y7	y6	y5	y4	y3	y2	y1	$\overline{y0}$	0	0	0	0	0	0	1
y7	y6	y5	y4	y3	y2	y1	y0	0	0	0	0	0	0	0

Если имеются отличия в старших разрядах d7 и y7 ($d7=\overline{y7}$) «нового» D и «старого» Y кодов (см. первую строку таблицы), то независимо от соотношения бит в младших семи разрядах сравниваемых кодов (см. знаки «×» в таблице) на выходах Z логического блока формируются единичные сигналы. Это означает, что досрочного конца сеанса связи не будет. Аналогичная ситуация описывается второй строкой табл. 5.3: совпадение бит в старших разрядах есть, но микроЭВМ может узнать об этом только после считывания предыдущего разряда, а в этом случае уже не имеет значения, прямо или косвенно будет определено значение бита в старшем разряде.

Последующие строки табл. 5.3 описывают все более благоприятные ситуации, при которых увеличивается число совпадающих старших разрядов кодов D и Y. В соответствии с этим в коде Z крайний правый 0 (признак размыкания линии связи 3) продвигается в сторону младших разрядов кода (остальные нули в коде Z несущественны) и в конечном счете, при полном совпадении кодов, z0=0 (см. последнюю строку табл. 5.3), так что сеанс связи оказывается вообще невозможным.

В примере, приведенном на рис. 5.16, новый и старый коды (D и Y) совпадают по четырем старшим разрядам (см. строку табл. 5.3, помеченную знаком « * »), а в разрядах d3 и y3 имеются отличия ($d3=\overline{y3}$). В результате в регистр RG3 заносится код 0001111, который по мере сдвига полезной информации в регистре RG1 также сдвигается вниз. После передачи в микроЭВМ четырех младших разрядов кода X производится очередной сдвиг информации в регистрах RG1 и RG3, в результате в нижний разряд регистра RG3 попадает лог. 0 (W=0), размыкающий линию связи 3. Сеанс связи при этом заканчивается досрочно, без передачи оставшихся четырех бит кода X.

Двоичный 5-разрядный счетчик СТ «подсчитывает» число сеансов связи, начиная счет с нуля при переполнении. Прибавление единицы к содержимому счетчика производится после окончания сеанса связи, при переходе сигнала S из 0 в 1. Если в счетчике зафиксирован код 11111_2, то срабатывает 5-входовой элемент И — НЕ, подключенный к его выходам. Сигнал лог. 0 с выхода элемента И — НЕ поступает на вход трехвходового элемента И и запрещает размыкание линии 3 сигналом W=0, если он будет сформирован. Таким образом, по крайней мере каждый 32-й сеанс связи оказывается полным, выполняемым без прерывания тока блоком 1.

В исходном состоянии в регистрах RG1, RG2, RG3 и счетчике СТ записана произвольная информация. Установление правильного начального состояния системы, как отмечалось, производится при выполнении не более 32 циклов чтения кода X. Так как в установившемся режиме число следующих друг за другом сокращенных сеансов связи не может превышать 31, микроЭВМ способна обнаруживать ошибки, связанные с нарушением этого условия.

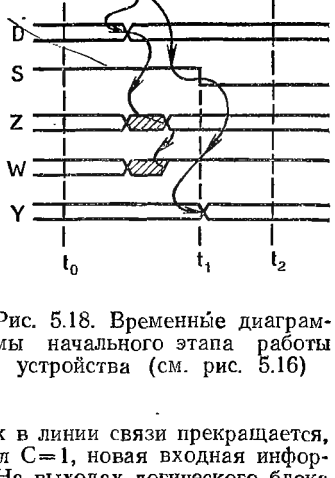


Рис. 5.18. Временны́е диаграммы начального этапа работы устройства (см. рис. 5.16)

Рассмотрим подробнее начало сеанса связи. В исходном состоянии A=0, в линии связи 3 протекает ток i0 или i1 в зависимости от значения сигнала Q. В момент t_0 (рис. 5.18) микроЭВМ начинает сеанс связи выдачей сигнала A=1. Ток в линии связи прекращается, поэтому в блоке 1 формируется сигнал C=1, новая входная информация запоминается в регистре RG1. На выходах логического блока L формируется код Z, который проходит через регистр RG3, сигнал W принимает установившееся значение (лог. 1). На рис. 5.18 участки временны́х диаграмм, соответствующие переходным процессам, заштрихованы.

Сигнал C=1 вызывает формирование сигнала S=0, по отрицательному фронту которого (момент t_1) регистр RG3 закрывается по входам и переходит в режим сдвига, а регистр RG2 принимает информацию из регистра RG1. (Эта информация в следующем сеансе связи будет рассматриваться как «старая».) В момент t_2 микроЭВМ формирует сигнал A=0, разрешая протекание тока в линии связи 3.

Далее (рис. 5.19) в период $t_3 — t_{10}$ процесс передачи информации через устройство полностью совпадает с рассмотренным ранее (см. рис. 5.8). Однако к моменту t_{11} W=0, передающие оптроны блока 1 выключены, поэтому B0=B1=1 и микроЭВМ принимает к сведению факт досрочного окончания сеанса связи. В момент t_{12} устройство готово к следующему сеансу связи.

При выполнении условия досрочного окончания сеанса связи можно уменьшить длительность отрицательного импульса S, чтобы приблизить момент готовности устройства к очередному сеансу связи. Однако это приводит к дальнейшему усложнению схемы блока 1.

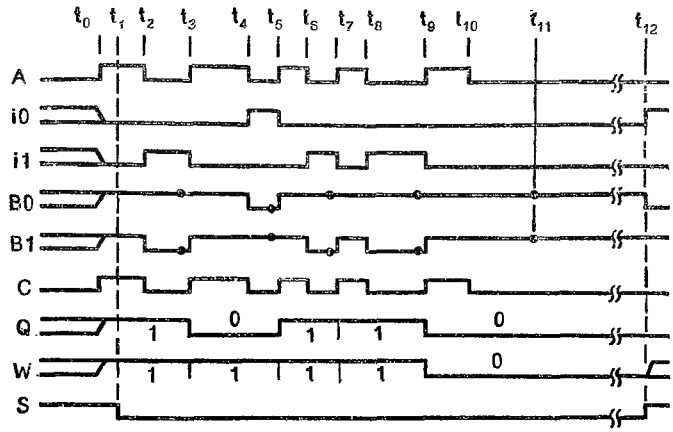


Рис. 5.19. Временны́е диаграммы передачи информации через устройство (см. рис. 5.16).

Сеанс связи заканчивается досрочно, в момент t_{11}

На рис. 5.20 показан пример подключения двух независимых источников информации к устройству. Аналоговые сигналы E1 и E2 от двух температурных датчиков с помощью аналого-цифровых пре-

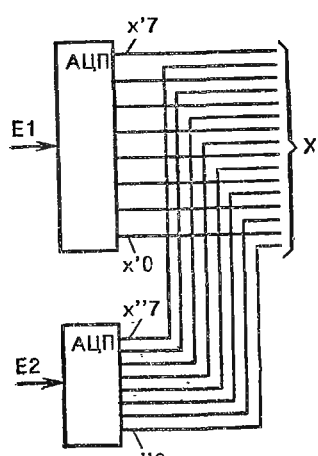


Рис. 5.20. Правильное подключение двух источников информации к устройству, показанному на рис. 5.16.

В коде X медленно меняющиеся разряды не должны чередоваться с быстро меняющимися

образователей (АЦП) преобразуются в коды x′7, x′6, ..., x′0 и x″7, x″6, ..., x″0 (x′7 и x″7 — старшие разряды), причем последний «вкладывается» в первый, в результате образуется код X=x′7, x″7, x′6, x″6, ..., x′0, x″0, который подается на входы 16-разрядного регистра RG1 блока 1 устройства. В полученном результирующем коде медленно меняющиеся (старшие) разряды обоих кодов сведены в единую группу, относительно стабильную во времени, что позволяет эффективно использовать сокращенные сеансы связи.

5.2. Схема связи микроЭВМ с мощным потребителем энергии

На рис. 5.21, *а* показана схема, позволяющая с помощью микроЭВМ управлять средней мощностью, выделяемой в нагрузке, например в нагревательном элементе печи. Нагрузка подключается к сети переменного тока (220 В, 50 Гц) через тиристорную схему, управление которой осуществляется через оптронную развязку от D-триггера. Если триггер установлен в 0, то нагрузка отключена от сети. Если триггер установлен в 1, то нагрузка подключена к сети.

Выпрямленное пульсирующее напряжение (см. эпюру сигнала a на рис. 5.21, *а*) поступает на формирователь импульсов, с выхода которого снимаются импульсы, причем их фронты по возможности приближены к моментам прохождения сетевого напряжения через 0 (см. эпюру b). Положительные перепады напряжения в точке b вызывают запоминание сигнала с выхода компаратора в D-триггере и прибавление единицы к текущему содержимому счетчика. При переполнении счетчика он начинает счет с нуля, затем через некоторое время он вновь переполняется и т. д.

Компаратор сравнивает n-разрядное положительное двоичное число A, поступающее со счетчика, с (n+1)-разрядным числом B, поступающим из программно-доступного порта микроЭВМ. Число B определяет среднюю мощность, которая должна выделяться в нагрузке, и остается стабильным до тех пор, пока не требуется изменить мощность. Если A≥B, то на выходе компаратора формируется лог. 1, в противном случае — лог. 0.

Управление мощностью, выделяемой в нагрузке, производится следующим образом. Чтобы полностью отключить нагрузку от сети, микроЭВМ записывает в выходной порт любое (n+1)-разрядное число, содержащее в старшем разряде единицу. При этом n-разрядные числа A, генерируемые счетчиком с частотой 100 Гц, оказываются меньшими, чем число B, и на выходе компаратора постоянно присутствует сигнал лог. 0, который каждый раз по фронту сигнала b переписывается в триггер, подтверждая его нулевое состояние, соответствующее отключенной нагрузке.

Чтобы нагрузка не отключалась от сети, микроЭВМ записывает в выходиой порт нулевой код (B=0). При этом условие A≥B выполняется для любого кода в счетчике, на выходе компаратора постоянно присутствует сигнал лог. 1, тиристорная схема включена.

При записи в выходной порт чисел B, лежащих в пределах $0<B\leqslant(2^n-1)$, обеспечиваются режимы дозированной подачи энергии в нагрузку. Пусть, например, n=5, а B=7. В этом случае счетчик вырабатывает непрерывную последовательность чисел 0, 1, 2, ..., 31, 0, 1, 2 и т. д., так что период повторения процессов, протекающих в схеме, равен 32 полупериодам сигнала в первичной питающей сети

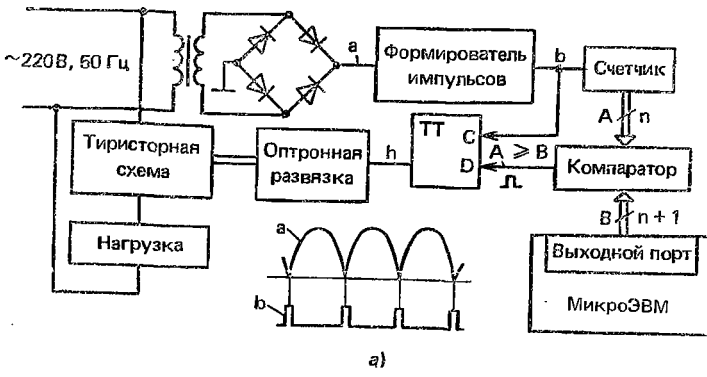


Рис. 5.21. Управление средней мощностью, выделяемой в нагрузке:
а — основная схема, *б—е* — модификации

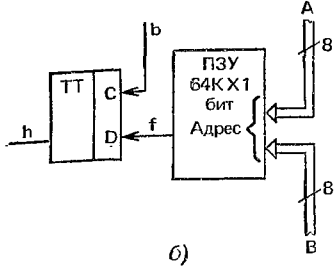
(320 мс). Тиристорная схема отключена в течение 7 полупериодов (при A=0, 1, 2, ..., 6) и включена в течение оставшихся 25 полупериодов (при A=7, 8, 9, ..., 31), т. е. в нагрузке выделяется около 78 % мощности по сравнению с максимальной. Таким образом, микроЭВМ может задавать 2^n градаций мощности, меняя числа B в выходном регистре.

Благодаря привязке импульсов b к моментам отсутствия напряжения в питающей сети обеспечивается «плавность» включения нагрузки. МикроЭВМ может менять числа B в любые моменты, не заботясь о ситуации в питающей сети, — привязка к нулю сетевого сигнала производится с помощью D-триггера.

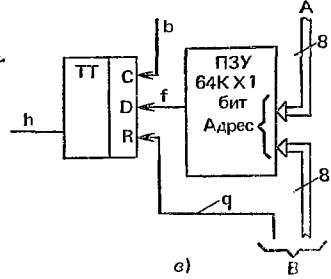
Если подключать нагрузку к сети в произвольные моменты, то рано или поздно появятся особо неприятные ситуации, когда момент включения приходится на вершину положительной или отрицательной полуволны переменного сетевого напряжения. (Иногда в таких случаях очень некстати перегорают лампочки накаливания в квартире.) Потребление тока в таких ситуациях будет иметь характер «большого скачка». Это особенно заметно, если нагрузка имеет емкостную составляющую, поскольку в момент включения емкость разряжена и можно в первом приближении считать, что она представляет собой короткое замыкание. «Удары» по цепям питания при скачкообразном изменении тока приводят к появлению в сети кратковременных импульсных помех, которые могут воздействовать на цифровые устройства (см. гл. 4).

Индуктивная нагрузка в момент подачи на нее скачка напряжения представляет собой бесконечно большое сопротивление и, следовательно, она не может являться источником помех по сети при включении. Другими словами, включение чисто индуктивной нагрузки можно было бы производить в произвольные моменты, не привязываясь к нулевым точкам синусоиды.

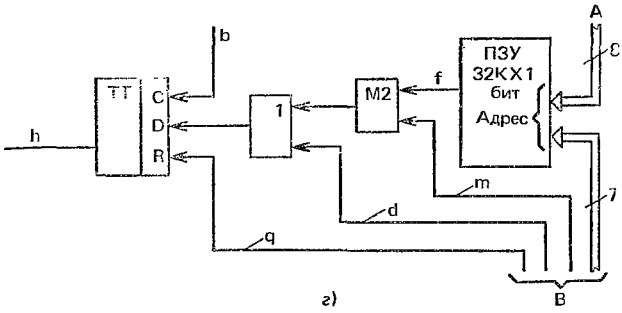
Интересно отметить, что при использовании тиристорных схем процесс выключения любой, в том числе индуктивной, нагрузки в произвольные моменты также протекает «бесшумно», хотя из-



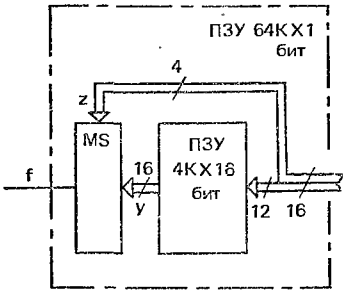
б)



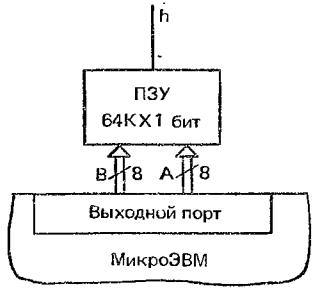
в)



г)



д)



е)

вестно, что невозможно скачкообразно изменить ток через индуктивность без катастрофических последствий. Здесь, однако, нет противоречия. Дело в том, что попытка прервать ток через тиристор на максимуме полуволны тока через индуктивность окажется безуспешной. Тиристор, обладая S-образной вольт-амперной характеристикой, «помнит», что он был включен (хотя по управляющей цепи уже поступил сигнал выключения), и его проводящее состояние будет сохраняться до тех пор, пока ток не снизится до величины, близкой нулю.

К сожалению, нет ни идеальных индуктивностей, ни идеальных тиристоров.

Теперь становится ясно, почему сетевой таймер представляется очень полезным устройством (см. § 1.2). Если такой таймер входит в состав микроЭВМ, то в приведенной на рис. 5.21, *а* схеме можно вход h оптронной развязки отключить от выхода D-триггера и подключить к одному из разрядов выходного порта микроЭВМ. Триггер и все сопутствующие ему элементы (начиная от трансформатора и кончая компаратором) исключаются из схемы, а функции счетчика и компаратора реализуются программно.

Правильно спроектированный сетевой таймер должен формировать импульсы, фронты которых предупреждают процессор о том, что через время, равное одному — трем десяткам микросекунд, синусоида питающего напряжения пройдет через нуль. По фронту такого импульса процессор выполняет стандартную процедуру прерывания по определенному вектору, т. е. временно откладывает выполнение текущего задания, запоминает параметры возврата и переходит к программе реакции на прерывание.

В нашем примере прерывающая программа предписывает процессору выполнить следующие действия:

1. Проверить содержимое ячейки памяти, в которой хранится число A (n-разрядный счетчик). Если $A \geqslant 31$ или $A < 0$, то записать нуль в эту ячейку, в противном случае — прибавить единицу младшего разряда к числу A.

2. Проверить условие $A \geqslant B$. (Число B определяет среднюю мощность, задается предварительно и хранится в другой ячейке памяти.) Если условие выполнено, выдать в разряд порта, связанный с оптронной развязкой, лог. 1, если условие не выполнено, выдать в этот разряд лог. 0.

3. Вернуться к прерванной задаче.

При включении напряжения питания микроЭВМ программа инициализации должна позаботиться о том, чтобы прерывания от сетевого таймера не воспринимались процессором без его санкции, а разряд порта, связанный с оптронной развязкой, был установлен в нулевое состояние.

Рассмотренные схемное и программное решения задачи управления выделяемой в нагрузке мощностью имеют общий недостаток. Он состоит в том, что нагрузка, в нашем примере нагревательный элемент печи, на протяжении полного цикла работы (при $n=5$ цикл составляет 320 мс) успевает только один раз побывать во включенном состоянии и один раз в выключенном. При $B=7$, как было показано, в течение 70 мс нагрузка отключена от сети, а в течение оставшихся 250 мс в ней выделяется мощность. Таким образом, нагревательный элемент «мерцает» с частотой около 3 Гц.

Частота мерцаний снижается с повышением разрешающей способности системы. Так, при $n=8$ число градаций мощности составляет 256, а длительность цикла работы возрастает до 2,56 с. В этом случае работа нагревательного элемента скорее напоминает даже не мерцания, а вспышки лампы маяка. Задача получения более равномерного по времени распределения выделяемой мощности решается с использованием дополнительных аппаратных или (и) программных средств.

Рассмотрим решение задачи на аппаратном уровне. На рис. 5.21, *б* показан фрагмент схемы, который нужно «подставить» на рис. 5.21, *а* (с учетом того, что разрядность чисел A и B одинако-

ва). Другими словами, вместо компаратора в даином случае используется ПЗУ со структурой 64К×1 бит. Восемь старших разрядов адреса ПЗУ поступают с выходного порта микроЭВМ и определяют один из $2^8=256$ уровней мощности, которая должна выделяться в нагрузке. Эта группа разрядов может рассматриваться как указатель страницы ПЗУ, содержащей 256 ячеек. Восемь младших разрядов адреса ПЗУ поступают с 8-разрядного двоичного счетчика. Эти разряды определяют позицию ячейки в пределах выбранной страницы. В процессе работы счетчика идет последовательный циклический перебор ячеек на выбранной странице. При этом с выхода ПЗУ выдаются сигналы включения нагрузки, более или менее равиомерно распределенные во времени (а не сгруппированные в «пачки», как при использовании компаратора).

Если нулевой код В соответствует полному отключению нагрузки, то на нулевой странице ПЗУ должна быть закодирована нулевая информация. В этом случае в процессе циклического перебора адресов счетчиком с выхода ПЗУ постоянно снимается сигнал лог. 0 и тиристорная схема поддерживается в выключенном состоянии.

Если, например, $B=67_{10}$, то в нагрузке будет выделяться мощность, которая составляет $67/256\approx0{,}26$ максимальной. При этом на странице ПЗУ с номером $67_{10}=01000011_2$ будут размещены 67_{10} единиц, приблизительно равномерно «разбросанных» по этой странице. Фрагмент кодировки страницы 67 ПЗУ приведен в табл. 5.4.

Кодировка получена на основании следующих подсчетов. Разделив 256 на 67, получим 3 и 55 в остатке. Это означает, что в каждую третью ячейку ПЗУ (на странице с номером 67) следует для начала записать лог. 1. Начальный вариант задания кодировки ПЗУ приведен в соответствующем столбце табл. 5.4. Предположив, что такая плотность записи единиц (в одной ячейке из трех записана лог. 1, а в двух последующих — лог. 0) распространяется на всю страницу, получим, что общее число единиц на странице окажется равным 85. Действительно, разделив 256 на 3, получим, что страница состоит из 85 полных и одной неполной триады. Неполная триада содержит одну ячейку, в которой, как предполагается, хранится лог. 0. Адрес этой ячейки для определенности примем равным 17407_{10}.

Теперь задача состоит в «прореживании» единиц, а именно в удалении лишних 18 единиц из имеющихся 85. Разделив 256 на 18, получим 14 и 4 в остатке. Не обращая внимания на остаток, заключаем из этого, что страницу нужно условно поделить на группы, в каждой из которых 14 ячеек, и из каждой такой группы удалить одну единицу. Таких групп окажется 18, и мы, таким образом, удалим 18 единиц из 85, т. е. оставим на странице 67 единиц (см. столбец табл. 5.4, отведенный для промежуточного результата кодировки; стрелками показан процесс уничтожения лишних единиц).

Для получения окончательной кодировки остается лишь несколько упорядочить множество имеющихся единиц, чтобы максимальная пауза при подаче порций энергии в нагрузку не превышала трех полупериодов сетевого питающего напряжения (каждый бит на странице соответствует одному полупериоду). В последнем столбце табл. 5.4 показано окончательное распределение единиц. Стрелки соответствуют «пересадке» единиц в пустующие области. Задача решена.

Если проинвертировать полученную (окончательную) кодировку, то можно получить «обратный» режим, при котором выделяемая

Таблица 5.4

Адрес ПЗУ			Выходной сигнал f		
Старшие разряды	Младшие разряды	Полный адрес	начальный	промежуточный	окончательный
01000011	00000000	17152	1	1	1
01000011	00000001	17153	0	0	0
01000011	00000010	17154	0	0	0
01000011	00000011	17155	1	0	0
01000011	00000100	17156	0	0	1
01000011	00000101	17157	0	0	0
01000011	00000110	17158	1	1	0
01000011	00000111	17159	0	0	0
01000011	00001000	17160	0	0	1
01000011	00001001	17161	1	1	0
01000011	00001010	17162	0	0	0
01000011	00001011	17163	0	0	0
01000011	00001100	17164	1	1	1
01000011	00001101	17165	0	0	0
01000011	00001110	17166	0	0	0
01000011	00001111	17167	1	1	1
01000011	00010000	17168	0	0	0
01000011	00010001	17169	0	0	0
01000011	00010010	17170	1	0	0
01000011	00010011	17171	0	0	1
01000011	00010100	17172	0	0	0
01000011	00010101	17173	1	1	0
01000011	00010110	17174	0	0	0
01000011	00010111	17175	0	0	1
01000011	00011000	17176	1	1	0
01000011	00011001	17177	0	0	0
01000011	00011010	17178	0	0	0
01000011	00011011	17179	1	1	1
01000011	00011100	17180	0	0	0
. . .	. . .	...	. . .	. . .	. . .
01000011	11111001	17401	1	1	1
01000011	11111010	17402	0	0	0
01000011	11111011	17403	0	0	0
01000011	11111100	17404	1	1	1
01000011	11111101	17405	0	0	0
01000011	11111110	17406	0	0	0
01000011	11111111	17407	0	0	0

в нагрузке мощность составит не 0,26, а 1—0,26=0,74 максимальной. В этом случае странице с номером 67 будет соответствовать «обратная» или «негативная» страница с номером 255—67=188. Аналогично полному отключению нагрузки (B=0) соответствует ее постоянное включение (страница с номером 255 содержит 256 единиц); режим, при котором B=1, соответствует режиму, при котором B=254, и т. д. Таким образом, кодировка разрабатывается для первых 128 страниц ПЗУ (с нулевой по 127-ю), а оставшиеся 128 стра-

ниц являются их обратными копиями. Разработка кодировки страницы, как было показано, сводится к выполнению цепи арифметических и логических операций и легко поддается автоматизации.

Фрагмент схемы, показанный на рис. 5.21, *б*, с рассмотренной кодировкой ПЗУ имеет следующую особенность. Ряд градаций мощности имеет разрыв в середине:

0, 1/256, 2/256, ..., 126/256, 127/256, 129/256, 130/256, ..., 254/256, 255/256, 1.

Чтобы ликвидировать разрыв, можно изменить кодировку ПЗУ и воспользоваться фрагментом схемы, показанным на рис. 5.21, *в*. Этот фрагмент, как и предыдущий, должен быть «подставлен» в схему, приведенную на рис. 5.21, *а*. Изменения в кодировке таковы. Страница с номером 0 на этот раз содержит одну единицу, страница с номером 1 — две единицы, страница с номером i содержит i+1 единицу. На последней, 255-й странице присутствуют 256 единиц. Эта структура также обладает симметрией: странице с номером 0 соответствует страница с номером 254, странице 1 — страница 253 и т. д. Страница с номером 127 теперь содержит 128 единиц и не имеет «зеркального» аналога.

Для полного отключения нагрузки от сети микроЭВМ устанавливает D-триггер в 0 выдачей статического сигнала лог. 1 в линию q. Момент установки триггера в 0 может быть произвольным, поскольку, как отмечалось, включенный тиристор не воспринимает сигнал выключения до тех пор, пока нагрузочный ток не снизится до величины, близкой нулю. (Установка триггера в 1 должна быть обязательно привязана по времени к моменту отсутствия напряжения в первичной питающей сети.) Если q=0, то микроЭВМ имеет возможность выбора одной из 256 страниц ПЗУ, в которых реализована непрерывная и равномерная шкала мощностей: 1/256, 2/256, 3/256, ..., 255/256, 1.

В силу симметрии кодировки ПЗУ его емкость можно сократить в два раза, вводя последовательно включенные элементы: сумматор по модулю два (элемент Исключающее ИЛИ) и элемент ИЛИ (рис. 5.21, *г*). При этом ПЗУ содержит 128 страниц, по 256 ячеек на странице. На нулевой странице записана одна единица, на первой — две и т. д., на 127-й — 128 единиц.

Для поддержания нагрузки в отключенном состоянии микроЭВМ постоянно поддерживает в линии q сигнал лог. 1 (отключение может производиться в произвольный момент). Во всех остальных случаях в линии q постоянно присутствует сигнал лог. 0. В режиме максимальной мощности из микроЭВМ в линию d постоянно подается сигнал лог. 1, который проходит на информационный вход D-триггера и обеспечивает поддержание его в состоянии лог. 1. Во всех остальных случаях в линии d постоянно присутствует сигнал лог. 0. При всех промежуточных градациях мощности вступает в действие ПЗУ, выходной сигнал f которого при m=0 проходит на вход D-триггера без инвертирования, а при m=1 — с инвертированием. В зависимости от сигнала m микроЭВМ обращается либо к «основной», либо к несуществующей «негативной» странице. При работе со 127-й страницей состояние сигнала m не имеет значения.

На рис. 5.21, *д* приведен пример использования ПЗУ со структурой 4К×16 бит в качестве ПЗУ со структурой 64К×1 бит. Преобразование структур достигается включением в схему мультиплексора MS, который в зависимости от кода на управляющих линиях z подключает к выходу один из входов y.

Все рассмотренные схемные решения могут быть реализованы программно, если в состав микроЭВМ входит сетевой таймер. В этом случае, как отмечалось, вход h оптронной развязки изолируется от выхода D-триггера и подключается непосредственно к одному из разрядов выходного порта микроЭВМ. Опираясь на фиксированное значение B и текущее значение A, процессор каждый раз отыскивает в ПЗУ, входящем в состав микроЭВМ, нужный бит и передает его в выходной порт.

Недостаток такого решения состоит в том, что ПЗУ переносится внутрь микроЭВМ, а точнее, — в ее адресное пространство, где, как правило, и без того тесно. Поэтому, возможно, полезным окажется компромиссное решение, при котором ПЗУ остается снаружи, а микроЭВМ, снабженная сетевым таймером, выполняет лишь функции задания B и циклического подсчета полупериодов напряжения питания (помимо прочих функций, не имеющих отношения к нашей задаче). Схема подключения ПЗУ к микроЭВМ показана на рис. 5.21, *е*.

Входы ПЗУ в данном случае подключаются непосредственно к выходному порту микроЭВМ, из которого поступает константа B, определяющая уровень мощности, и переменная A — текущий результат циклического подсчета полупериодов. Выход ПЗУ подключается непосредственно к входу оптронной развязки. Схема, показанная на рис. 5.21, *е*, имеет те же недостатки, что и схема, показанная на рис. 5.21, *б*, поэтому здесь также возможны улучшения, аналогичные приведенным на рис. 5.21, *в*, *г*. Вместо D-триггера при этом используется двухвходовый элемент И, на который следует подать сигналы, ранее поступавшие на D- и R-входы триггера.

В некоторых случаях при кодировании ПЗУ нужно следить за тем, чтобы единицы распределялись в среднем равномерно по четным и нечетным адресам. Это позволит уравнять число положительных и отрицательных полуволн тока, протекающего через нагрузку, или, другими словами, предотвратить появление постоянной составляющей тока в цепи первичного питания.

Если вместо двоичного счетчика (рис. 5.21, *а*) использовать генератор псевдослучайных чисел (сдвиговый регистр с элементом Исключающее ИЛИ в цепи обратной связи), то проблема «прореживания единиц» будет решаться автоматически, так как поток чисел, поступающих на компаратор, будет не монотонно возрастающим, а «хаотическим» и компаратор будет формировать не ступенчатый сигнал, а множество «мелких» импульсов разной длительности.

Если подключить к точке h стрелочный вольтметр постоянного тока с достаточной инерционностью стрелочного механизма, то мы получим возможность визуальной регистрации (в аналоговой форме, по шкале вольтметра) величины B. В этом случае «побочным эффектом» работы системы является выполнение ею функций цифро-аналогового преобразования.

5.3. Передача запросных и ответных сигналов прерывания по двухпроводной линии связи [29]

Схема связи микроЭВМ с удаленными абонентами приведена на рис. 5.22. В исходном состоянии транзистор VT1 передающего оптрона микроЭВМ насыщен благодаря действию сигнала $\bar{d}=1$, представленного световым потоком от светодиода этого оптрона (свето-

диоды передающих оптронов и фотоприемники приемных оптронов не показаны). Запросов b от абонентов нет, транзисторы VT2 передающих оптронов выключены, ток по цепи источника U отсутствует, светодиоды приемных оптронов не излучают свет, т. е. a=0, c=0.

При появлении запроса от одного абонента (в отсутствие конкурентов) соответствующий транзистор VT2 включается, что приводит к включению всех приемных оптронов, расположенных справа от этого транзистора. Сигналы a=1 запрещают формирование новых запросных сигналов b в абонентских блоках, расположенных справа

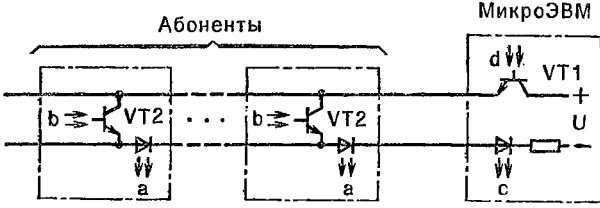


Рис. 5.22. Схема оптронного канала связи микроЭВМ с удаленными абонентами

от активного абонента. Блоки, расположенные слева от него, шунтируются транзистором VT2. Таким образом, «конкурентная борьба» возможна в достаточно коротком интервале времени, который определяется скоростью срабатывания блокирующих цепей (цепей запрета формирования новых запросных сигналов b=1 при наличии тока в линии). После окончания переходных процессов абонентский блок переходит в одно из трех устойчивых состояний:

1) a=0, b=0 или b=1; эти условия выполняются в блоках, расположенных слева от «выигравшего»;

2) a=1, b=0; эти условия характеризуют блоки, расположенные справа от «выигравшего»;

3) a=1, b=1; эти условия выполняются только в одном — «выигравшем» блоке.

Получив сигнал c=1, микроЭВМ выжидает определенное время, достаточное для завершения конкурентной борьбы (например, 5 мкс), и выключает транзистор VT1. Прекращение тока воспринимается выигравшим блоком как предупреждение о предстоящем сеансе связи, поэтому он подключается к общей информационной линии (не показана) и готовится к обмену. В дальнейшем после окончания обмена или в процессе обмена вновь включается транзистор VT1 и микроЭВМ вновь оказывается способной принять обобщенный сигнал прерывания (c=1).

5.4. Передача вектора в процессор, не обладающий способностью реакции на векторные прерывания [42]

Простейшие микропроцессоры не обладают способностью реагировать на так называемые векторные прерывания (см. гл. 2), а имеют лишь единственный вход прерывания, воздействие на кото-

рый вызывает передачу управления в фиксированную ячейку памяти с упрятыванием в стек информации, необходимой для последующего возврата к прерванной задаче. Методы поиска источников прерывания для таких процессоров уже рассматривались в § 2.5. Здесь приводится еще одна схема, позволяющая повысить «интеллект» процессора до уровня, при котором он способен различать поступающие запросы и соответствующим образом на них реагировать.

Схема передачи показана на рис. 5.23. При получении хотя бы одного запроса на выходе ИЛИ приоритетного шифратора формируется сигнал, вызывающий переход процессора к фиксированной пре-

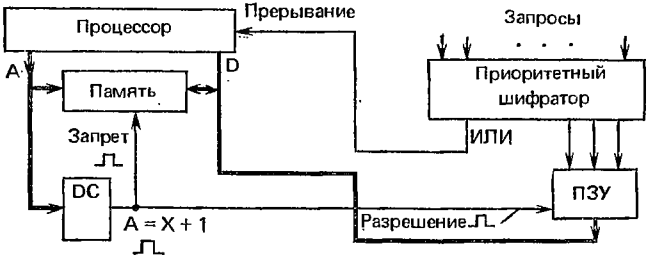


Рис. 5.23. Схема микроЭВМ с «принудительной» передачей вектора в процессор

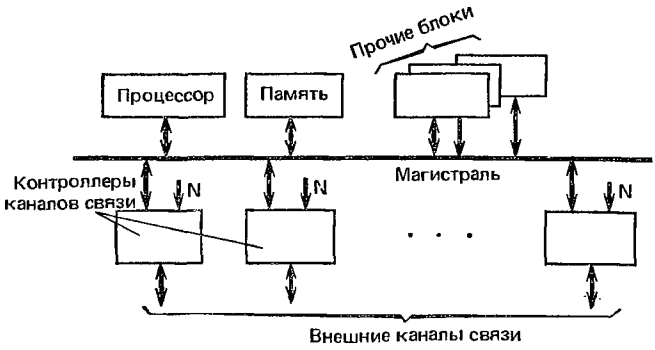
рывающей программе. В начале этой программы, а точнее в ячейках X и X+1, записана двусловная команда безусловной передачи управления по абсолютному адресу A′, записанному в ячейке X+1:

X	КОП
X+1	A′

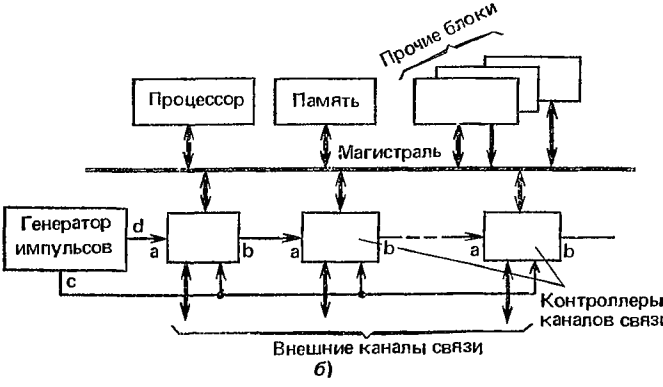
При попытке выбора адреса A′ из ячейки X+1 происходит следующее. Дешифратор DC, обнаружив, что текущий адрес равен X+1, выдает сигнал запрета в память, отключая ее от шины данных, и подключает к шине данных выход ПЗУ, в котором хранятся адреса начал всех программ обслуживания. Поскольку приоритетный шифратор выбрал наиболее «важный» запрос, с выхода ПЗУ поступает соответствующий код, который используется процессором для передачи управления. Процессор при этом «не подозревает о подлоге» и невольно отрабатывает процедуру перехода к нужной программе, т. е. фактически ведет себя так же, как и при отработке векторного прерывания. На схеме для простоты не показаны элементы синхронизации, препятствующие возникновению гонок.

5.5. Минимизация числа внешних связей устройств при задании их собственных адресов [114]

Для того чтобы однотипные устройства, например контроллеры внешних каналов связи, отличались друг от друга с точки зрения процессора, они должны иметь разные собственные адреса. Обычно для задания собственных адресов используются группы выводов N (рис. 5.24, *а*). Если, например, число выводов в группе равно шести, то к общей магистрали можно подключить до $2^6=64$ однотипных устройств, отличающихся «распайкой» кодов на входах N. В этом случае, например, «распайка» кода 000000_2 будет означать, что внутренние регистры устройства имеют адреса 177060_8 и 177062_8,



а)



б)

Рис. 5.24. Варианты подключении ряда однотипных контроллеров к микроЭВМ:

а — традиционный; *б* — предлагаемый

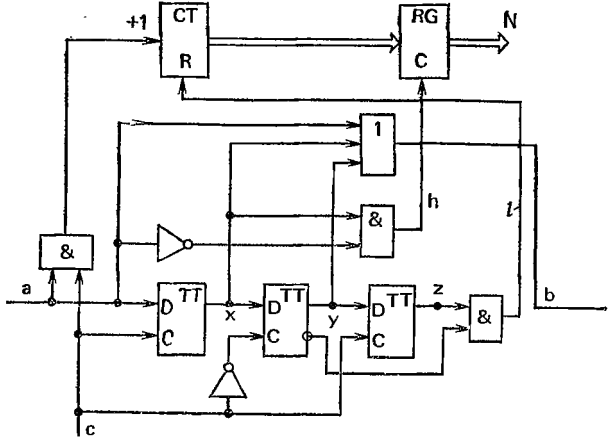


Рис. 5.25. Фрагмент схемы контроллера, показанного на рис. 5.24, *б*

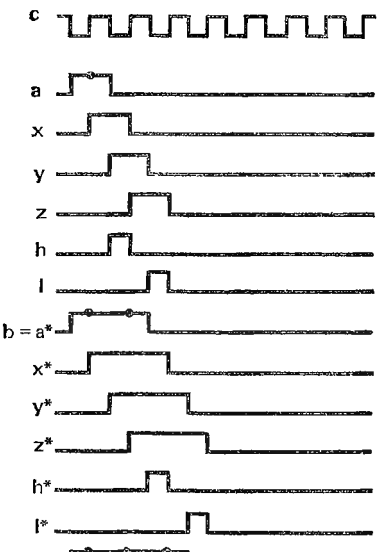


Рис. 5.26. Последовательное расширение передаваемого импульса

а вектор прерывания при обращении устройства к процессору равен 200_8. Если задать на входах N код 000001, то адреса и вектор сместятся относительно предыдущих и будут равны соответственно 177064, 177066 и 204 и т. д.

Предлагается сократить число выводов, предназначенных для задания собственного адреса устройства (рис. 5.24, *б*). В систему введен генератор импульсов, выход с которого подключен ко всем контроллерам параллельно, а выход d — к входу a первого контроллера. Далее сигнал (положительный импульс) с выхода d генератора проходит по цепочке контроллеров, каждый раз расширяясь на один период синхроимпульсов, снимаемых с выхода с генератора. Каждый контроллер в соответствии с длительностью импульса, поступающего на его вход a, определяет свою позицию (номер) в группе. Измеренное значение длительности (единица измерения — период сигнала

на входе c) хранится в регистре RG (рис. 5.25) и представляет собой номер контроллера. Ближайший к генератору импульсов контроллер вмеет номер 1, следующий — 2, следующий — 3 и т. д.

На вход a первого контроллера поступает положительный импульс, длительность которого равна одному периоду синхросигнала на входе c (рис. 5.26). Сигналы x, y я z представляют собой исходный сигиал a, сдвинутый по времени на половину периода, период

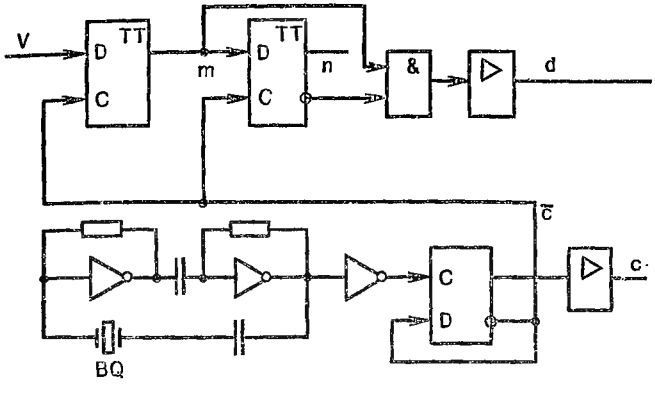


Рис. 5.27. Вариант построения генератора

и полтора периода. По сигналу h производится перепись накопленной в счетчике суммы в регистр. По сигналу на входе R счетчика производится его установка в 0. Из диаграммы a видно, что счетчик за время присутствия единичного сигнала иа входе a контроллера успевает накопить только одну единицу (см. точку на вершине импульса). Из диаграммы b видно, что импульс, пройдя через первый контроллер, расширяется на один период синхросигнала.

Временные диаграммы, соответствующие второму по порядку контроллеру, имеют те же обозначения, но помечены знаками *.

Длительность импульса b составляет два периода синхросигнала, следовательно, счетчик во втором контроллере накапливает две единицы (см. две точки на вершине импульса b) и т. д. Поскольку импульсы h и l разнесены во времени, регистр выполняется на одно-каскадных триггерах с потенциальными входами синхронизации (в отличие от D-триггеров, имеющих двухъярусную внутреннюю структуру).

Схема генератора импульсов приведена на рис. 5.27. Низкочастотный сигнал V, например, частотой 50 Гц определяет частоту генерации импульсов d, подаваемых на цепочку контроллеров, и может сниматься с блока питания. Высокочастотный сигнал вырабатывается стандартной кольцевой схемой с использованием кварцевого резонатора BQ в цепи обратной связи.

После включения напряжения питания и установления стабильного режима работы генератора импульсов в регистры (см. рис. 5.25) всех контроллеров записываются их номера: 1, 2, 3, ... и т. д. в порядке удаления от генератора. Эти номера затем постоянно (с частотой 50 Гц) перевычисляются, что, однако, не нарушает стабильности кодов в регистрах. Такой механизм назначения собственных адресов устройств (памяти, контроллеров и т. п.) требует всего трех выводов в каждом устройстве независимо от их числа в цепочке. Число выводов может быть сокращено до двух при использовании, например кода «Манчестер-II», (см. гл. 3) при некотором усложнении внутренних схем.

5.6. Определение состояния переключателя без использования схемных элементов опроса [43]

На рис. 5.28 показана схема, содержащая центральный процессор ЦП, адаптер связи с последовательными каналами, блок сопряжения и два периферийных устройства — телетайп и дисплей (память и другие устройства не показаны как несущественные для понимания идеи). Пользователь с помощью переключателя SA сообщает процессору, с каким периферийным устройством предстоит работать. Это связано с тем, что программы обслуживания этих устройств

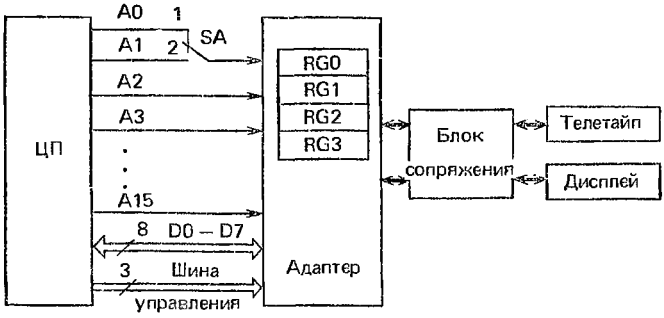


Рис. 5.28. Схема связи процессора с периферийными устройствами

разные и процессор должен «узнать», в какое положение установлен переключатель.

Если переключатель установлен в положение 1, то в качестве младшего разряда адреса, поступающего в адаптер, будет использоваться младший разряд A0 адреса, поступающего из процессора. Следующий разряд A1 не поступает в адаптер, и, таким образом, адаптер «не различает» адреса, например, равные 8004_{16} и 8006_{16}, и расценивает их как относящиеся к одному и тому же регистру из группы RG0 — RG3. Поэтому процессор (программист) может поступить так.

1. Дать сигнал общего сброса, очистив регистры RG0—RG3.
2. Записать некоторое ненулевое число по адресу 8004.
3. Прочитать число по адресу 8006. Если записываемое и прочитанное числа совпали, то переключатель находится в положении 1, в противном случае — в положении 2.

5.7. Использование двухвекторных источников прерываний для демократизации их обслуживания [44]

В системе с поразрядным отбором претендентов на обслуживание со стороны центрального процессора (см. § 2.5, 3.2) каждый контроллер может являться источником прерывания ЦП по определенному вектору. Например, контроллер 4 способен генерировать вектор 250_8, контроллер 12 — вектор 120_8 и т. д. При наличии конкурирующих контроллеров победителем оказывается тот, у которого код вектора имеет большее численное значение. В данном примере победит первый из двух контроллеров и ЦП воспримет вектор 250. В ячейке 250 ОЗУ или ПЗУ процессор находит адрес первой команды программы обслуживания контроллера 4 (например, код 10AE) и переходит к выполнению этой программы, предварительно «упрятав» в стек параметры возврата к прерванной задаче.

В данном примере контроллер 12 всегда «оттесняется» контроллером 4, т. е. исход борьбы заранее определен. Если число контроллеров достаточно велико и между ними существует жесткий приоритетный порядок, то у самого «слабого» из них шансы на обслуживание могут оказаться близкими нулю.

Предположим теперь, что контроллер 4 вступает в борьбу с контроллером 12, имея в своем распоряжении либо вектор 250 (основной), либо вектор 127 (дополнительный). Выбор того или иного вектора, например, равновероятен. На протяжении одного сеанса конкурсного отбора вектор остается постоянным, а число участников конкурса не меняется. Предположим далее, что контроллер 12, в свою очередь, предъявляет на конкурс либо вектор 120 (с вероятностью 0,9), либо вектор 257 (с вероятностью 0,1). Дополнительные векторы в этом примере получаются путем инвертирования основных векторов. В двух случаях из четырех с общей вероятностью 0,9 выигрывает контроллер 4 (комбинации 250—120 с вероятностью $0{,}5 \cdot 0{,}9 = 0{,}45$ и 127—120 с вероятностью $0{,}5 \cdot 0{,}9 = 0{,}45$). В оставшихся двух случаях с общей вероятностью 0,1 выигрывает контроллер 12 (комбинацин 250—257 с вероятностью $0{,}5 \cdot 0{,}1 = 0{,}05$ и 127—257 с вероятностью $0{,}5 \cdot 0{,}1 = 0{,}05$). Таким образом, контроллер 4 сохраняет приоритет над контроллером 12, но периодически, в среднем в 10 % случаев, уступает ему возможность воспользоваться обслуживанием со стороны ЦП.

Поскольку каждый контроллер генерирует два разных вектора, а программа обслуживания одна и та же, то оба вектора должны отсылать ЦП в одну и ту же точку — в начало этой программы. Так, в нашем примере в ячейках ОЗУ или ПЗУ с адресами 250 и 127 должен храниться один и тот же код 10АЕ — адрес начала программы обслуживания контроллера 4. Аналогично в ячейках 120 и 257 должен храниться адрес начала программы обслуживания контроллера 12.

Если все контроллеры вычислительной системы наделены свойством генерации двух или более векторов, то эта система при надлежащем выборе спектров изменения векторов теряет жесткость и становится демократичной.

Рассмотрим вопрос о выборе непересекающихся множеств векторов прерывания. Для определенности будем считать, что имеются восемь контроллеров, каждый из которых может генерировать лишь два различных вектора. Так как в этом случае суммарное число различных векторов составляет $8 \cdot 2 = 16$, то разрядность кода, отображающего эту величину, должна быть равной или большей 4 — в последнем случае появляются дополнительные возможности по формированию спектров.

Далее для простоты предполагается, что разрядность равна 4. В каждом контроллере содержится схема генерации одного из двух альтернативных векторов, показанная на рис. 5.29. В регистре при-

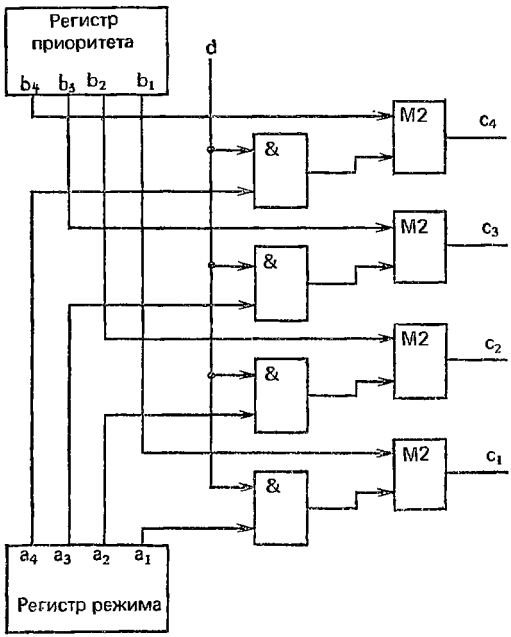


Рис. 5.29. Схема генерации вектора

оритета хранится основной вектор. При $d=0$ элементы И закрыты и на выходы c_4, c_3, c_2, c_1 транслируется код $b_4b_3b_2b_1$, т. е. основной вектор. При $d=1$ на выходы передается дополнительный вектор, который получается из основного путем инвертирования одного, нескольких или всех разрядов основного вектора в зависимости от кода в регистре режима. Единичные позиции кода $a_4a_3a_2a_1$ определяют позиции разрядов кода $b_4b_3b_2b_1$, подлежащие инвертированию. Например, при $d=1$, $a_4a_3a_2a_1=1011$, $b_4b_3b_2b_1=1001$ на выходах схемы будет сформирован код $c_4c_3c_2c_1=0010$, поскольку остается неизменным лишь значение разряда 3 входного кода $b_4b_3b_2b_1$, а остальные разряды инвертируются.

Таким образом, любое число из диапазона $[0—15]_{10}$ можно превратить в любое число из этого же диапазона, инвертируя некоторые (или все) его разряды. В приведенном примере число 9_{10} превращается в число 2_{10}. Сигнал d формируется генератором случайных чисел (не показан) с заданной вероятностью. На протяжении одного сеанса конкурсного отбора этот сигнал должен оставаться неизменным. Исходная информация для построения спектров с учетом принятых ограничений приведена в табл. 5.5.

Таблица 5.5

Код в регистре приоритета	Код в регистре режима															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	0	3	2	5	4	7	6	9	8	11	10	13	12	15	14
2	2	3	0	1	6	7	4	5	10	11	8	9	14	15	12	13
3	3	2	1	0	7	6	5	4	11	10	9	8	15	14	13	12
4	4	5	6	7	0	1	2	3	12	13	14	15	8	9	10	11
5	5	4	7	6	1	0	3	2	13	12	15	14	9	8	11	10
6	6	7	4	5	2	3	0	1	14	15	12	13	10	11	8	9
7	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8
8	8	9	10	11	12	13	14	15	0	1	2	3	4	5	6	7
9	9	8	11	10	13	12	15	14	1	0	3	2	5	4	7	6
10	10	11	8	9	14	15	12	13	2	3	0	1	6	7	4	5
11	11	10	9	8	15	14	13	12	3	2	1	0	7	6	5	4
12	12	13	14	15	8	9	10	11	4	5	6	7	0	1	2	3
13	13	12	15	14	9	8	11	10	5	4	7	6	1	0	3	2
14	14	15	12	13	10	11	8	9	6	7	4	5	2	3	0	1
15	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Все коды представлены в виде десятичных чисел. Таблица позволяет определять дополнительный вектор по известному основному вектору, записанному в регистр приоритета, и содержимому регистра режима. Пусть, например, в регистр приоритета одного из контроллеров записан код 14, а в регистр режима этого контроллера — код 7. Выбирая элемент таблицы на пересечении соответствующей строки и столбца, получаем код 9. Это означает, что при такой настройке данный контроллер способен генерировать два вектора: 14 и 9. Заметим, что те же векторы генерируются и при записи ко-

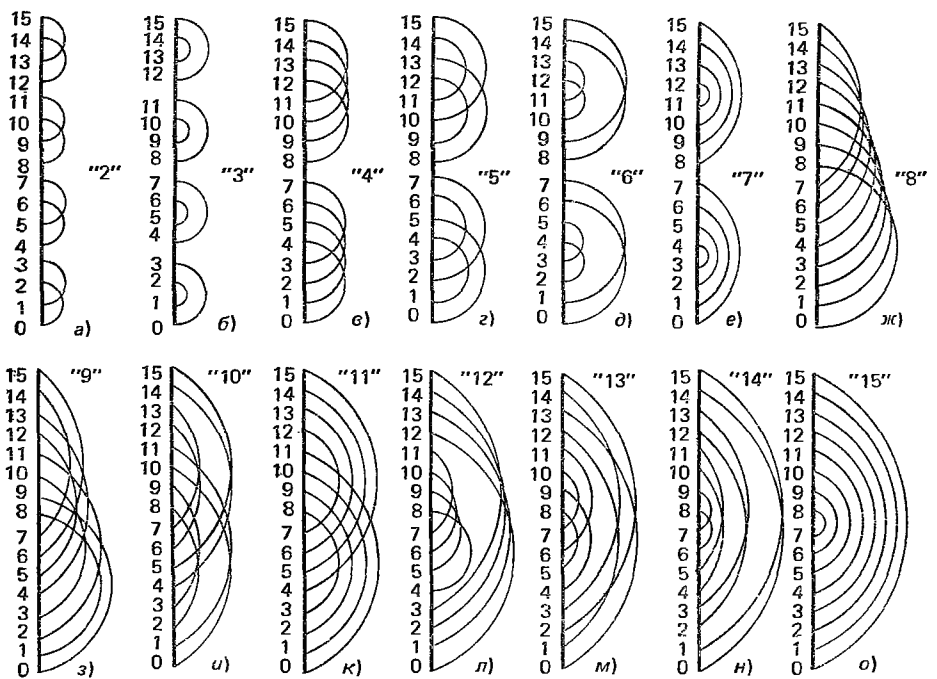


Рис. 5.30. Распределение пар векторов, генерируемых контроллерами

да 9 в регистр приоритета с сохранением кода 7 в регистре режима. Полученная пара векторов 9 и 14 закрепляется за данным контроллером и не может быть использована в других контроллерах в силу необходимости соблюдения условия однозначной идентификации контроллера центральным процессором.

Далее для простоты предполагается, что во всех контроллерах системы в регистры режима записан один и тот же код, в данном примере код 7. Использование разных кодов приводит к значительному увеличению числа способов задания спектров, и поэтому здесь не рассматривается. Указанное ограничение закрепляет в табл. 5.5 столбец, соответствующий коду 7 в регистре режима. При этом для оставшихся контроллеров выделяются следующие пары векторов: 0 и 7, 1 и 6, 2 и 5, 3 и 4, 8 и 15, 10 и 13, 11 и 12. Диаграммы, приведенные на рис. 5.30, *а — о*, дают наглядное представление о распределении пар векторов между контроллерами. Диаграмма, показанная на рис. 5.30, *а*, соответствует столбцу 2 табл. 5.5 (столбцы 0 и 1 не представляют интереса), на рис. 5.30, *б* — столбцу 3 и т. д., рис. 5.30, *о* соответствует столбцу 15 таблицы. Цифрами в кавычках на рис. 5.30 помечены коды в регистре режима, которые порождают соответствующие диаграммы.

Каждому контроллеру соответствует дуга, соединяющая основной и дополнительный векторы, представленные десятичными числами, размещенными в порядке возрастания на числовой оси. Рассмотренному примеру (столбец 7 табл. 5.5) соответствует диаграмма, показанная на рис. 5.30, *е*. Из этой диаграммы видно, что все восемь контроллеров делятся на две одинаковые группы: «верхнюю» и «нижнюю». Контроллеры верхней группы обладают безусловным приоритетом над контроллерами нижней группы, а внутри группы (при равновероятном выборе одного из двух векторов в каждом контроллере) наибольшим вероятностным приоритетом обладает тот контроллер, которому соответствует наиболее длинная дуга.

5.8. Ввод данных в микроЭВМ с кассетного магнитофона [45]

Схема связи микроЭВМ с бытовым кассетным магнитофоном приведена на рис. 5.31. В режиме воспроизведения сигналы с магнитной ленты усиливаются и проходят через выходной трансформатор Тр усилителя низкой частоты (конструктивно он расположен внутри магнитофона) и конденсатор в точку a. Логический 0 представлен одним периодом синусоиды с частотой 3 кГц, а логическая 1 — одним периодом синусоиды с частотой 1,5 кГц. По положительному фронту сигнала, прошедшего через триггер Шмидта ТН в точку b триггер ТТ устанавливается в 1. МикроЭВМ в ответ на сигнал $\overline{S}$ (лог. 0) устанавливает в 0 триггер ТТ сигналом Y. Программа ведет подсчет числа циклов ожидания между двумя последовательными сигналами $\overline{S}$. Если это число соответствует 350 мкс, то микроЭВМ «узнаёт», что принят 0; если же число циклов примерно вдвое больше, то предполагается, что принята 1.

Запись на магнитофон производится выдачей с выходного порта микроЭВМ сигналов, по форме аналогичных сигналам в точке b. Возможно, что при записи сигналы придется выдавать в инверсном виде. Это определяется конкретным типом магнитофона (число инвертирующих каскадов в каналах записи — воспроизведения может быть четным или нечетным).

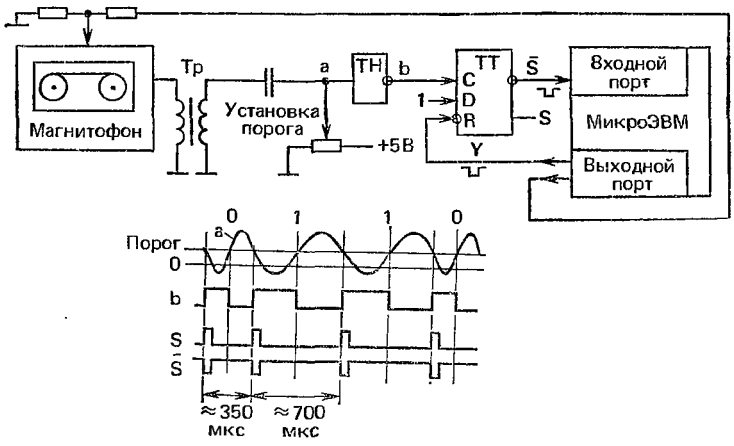


Рис. 5.31. Схема связи микроЭВМ с кассетным магнитофоном

5.9. Сопряжение микроЭВМ с клавиатурой и со стрелочными индикаторами с использованием общего порта ввода-вывода [46]

МикроЭВМ выдает результаты вычислений на стрелочные приборы (рис. 5.32). На выходах порта П2 присутствуют сигналы лог. 1 (напряжения высокого уровня), поэтому клавиатура «не мешает» работе порта П1, настроенного на передачу сигналов к стрелочным приборам (например, вольтметрам) через буферные элементы.

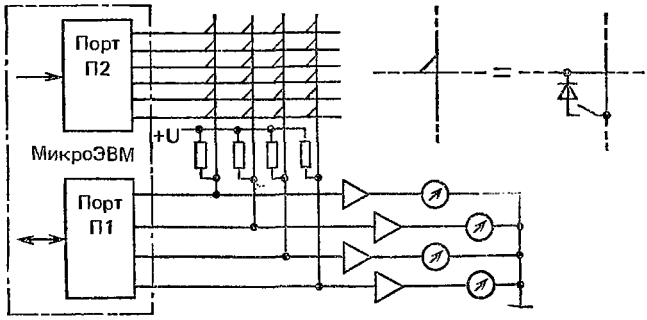


Рис. 5.32. Подключение клавиатуры и стрелочных индикаторов к микроЭВМ

Каждый стрелочный прибор усредняет импульсные сигналы с соответствующего канала. По мере увеличения степени «наполненности» канала импульсами (уменьшения скважности) показания стрелочного прибора в этом канале растут. Процессор, как было показано в гл. 2, способен управлять отдельными битами выходного порта, например, входящего в состав контроллера программируемого интерфейса. Поэтому он способен одновременно генерировать в разных каналах разные импульсные последовательности.

Для ввода информации с клавиатуры микроЭВМ на короткое время переключает двунаправленный порт П1 в режим приема, а на выходах порта П2 генерирует код «бегущий 0», опрашивая построчно клавиатурную матрицу и принимая полученные на каждом шаге данные (4 бита) через порт П1. Если ключ на пересечении выбранной строки и некоторого столбца замкнут, то напряжение на этом столбце снижается с +5 до +1 В. Если ключ разомкнут, то напряжение на столбце остается высоким. Осуществив цикл ввода, микроЭВМ вновь переключается на вывод. Если цикл ввода достаточно короток, то стрелочные приборы «не замечают» его. С увеличением отношения длительности цикла ввода к рабочему циклу ошибка измерения возрастает. Вместо механических ключей могут использоваться электронные ключи на транзисторах, тиристорах и т. п.

Эту схему можно упростить, если подключить группу стрелочных приборов с соответствующими буферными элементами к выходам порта П2, а не П1 (на рис. 5.32 эти изменения не отражены). Выходы порта П2 в этом случае могут быть выполнены либо по схеме с открытым коллектором с дополнительными нагрузочными резисторами между строками клавиатурной матрицы и шиной питания, либо по схеме с активным выходом, которая обеспечивает выдачу напряжения высокого или низкого уровня через «верхний» или «нижний» выходной транзистор.

Упрощение состоит в том, что порт П1 при таком подключении приборов может быть не двунаправленным, а однонаправленным, обеспечивающим только прием информации извне. (Многие микроЭВМ имеют только однонаправленные порты — входные и выходные, по своей структуре более простые, чем двунаправленные.)

Работа системы в этом случае, как и в предыдущем, сопровождается выдачей различных импульсных последовательностей на входы буферных (развязывающих) элементов и усреднением пульсирующих напряжений стрелочными вольтметрами постоянного тока, которые по существу выполняют функции цифро-аналоговых преобразователей. Опрос клавиатуры также может выполняться путем периодической кратковременной посылки «бегущего нуля» по строкам матрицы с регистрацией на каждом шаге результатов опроса, поступающих в микроЭВМ через однонаправленный порт П1. Однако в данном случае, проявив определенную изобретательность, можно построить программу, которая так «расставляет» импульсы в каналах, что опрос клавиатуры становится (по крайней мере, частично) «побочным результатом» выполнения «основной» работы — цифро-аналогового преобразования.

Отсчет элементарных временных интервалов при формировании импульсных последовательностей производится по сигналам прерывания, поступающим в процессор от сравнительно высокочастотного таймера (например, частотой 1 кГц). С увеличением разрешающей способности цифро-аналогового преобразования все более значимой будет становиться проблема «дрожания стрелки» на краях диапа-

зона измерения. Она по своей сути близка проблеме «мерцания» нагревательного элемента (см. § 5.2), и при ее решении можно воспользоваться рассмотренными методами «улучшения» импульсных последовательностей.

5.10. Сопряжение микроЭВМ с клавиатурой и с группой датчиков с использованием общего входного порта [47]

Линии входного порта П2 микроЭВМ (рис. 5.33) подключены через резисторы (не показаны) к шине положительного питания, инверторы имеют выходы типа открытый коллектор. В режиме опроса клавиатуры микроЭВМ, как обычно, генерирует код «бегущий 0» по строкам матрицы и каждый раз считывает через порт П2 семиразрядный код, определяемый состоянием клавиш в соответствующей строке. При этом на выходе А порта П1 поддерживается лог. 0, диоды VD поддерживают инверторы в выключенном состоянии. В режиме ввода информации с входов X (например, от дискретных датчиков) микроЭВМ выставляет во всех разрядах выходного порта П1 сигналы лог. 1, диоды закрываются, и информация в инверсном виде поступает в порт П2.

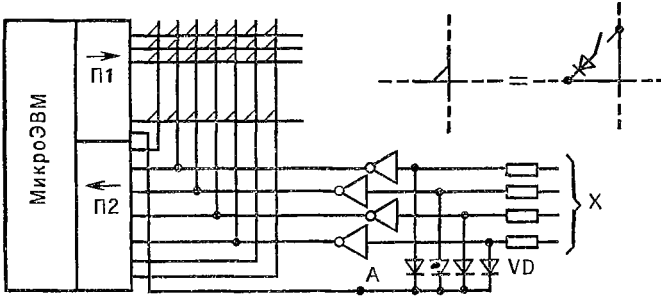


Рис. 5.33. Схема связи микроЭВМ с клавиатурой и группой датчиков

5.11. Подключение клавиатуры к магистрали микроЭВМ [48]

Использование принципа программной доступности (см. гл. 1) позволяет производить построчный опрос клавиатуры с выдачей результата непосредственно в шину данных микроЭВМ (рис. 5.34). Управляющие линии не показаны. Схема поглощает 16 ячеек адресного пространства.

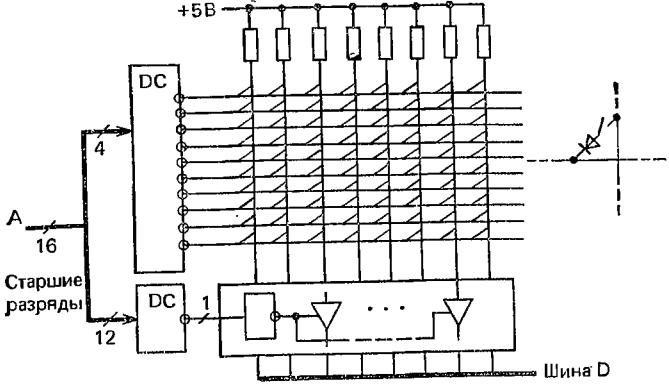


Рис. 5.34. Подключение клавиатуры к магистрали микроЭВМ

5.12. Подсветка нажатых клавиш ответными сигналами от микроЭВМ без использования дополнительных связей с клавиатурой [49]

В клавиатурной матрице (рис. 5.35) в качестве развязывающих диодов использованы светодиоды, подсвечивающие нажатые клавиши. (Если обратное допустимое напряжение светодиода мало, то можно использовать его в комбинации с двумя обычными диодами, см. рис. 5.3.) Подсветка выполняется не «механически», а свидетель-

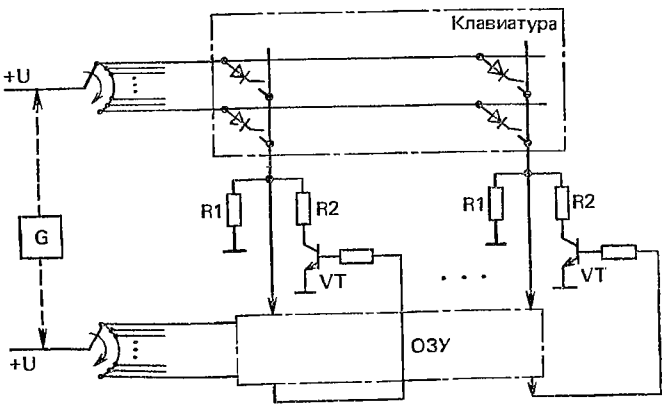


Рис. 5.35. Схема связи между клавиатурой и памятью микроЭВМ

ствует о том, что информация от клавиши записана в ОЗУ микроЭВМ. Отсутствие подсветки или неправильная подсветка позволяет оператору быстро определить факт ошибки (сбой или отказ) и предотвратить неправильную работу системы. С помощью генератора G ведется непрерывный последовательный опрос строк клавиатурной матрицы и синхроино с ним запись — чтение информации в — из ОЗУ.

Если при опросе строки обнаружится, что некоторый ключ замкнут, то ток от источника +U потечет через соответствующий светодиод и резистор R1 на «землю» (транзистор VT закрыт). Этот ток недостаточен для зажигания соответствующего светодиода, так как сопротивление резистора R1 выбрано большим. Логическая 1 (напряжение высокого уровня) будет записана в соответствующий разряд выбранной ячейки ОЗУ и тут же считана из него. При этом откроется транзистор VT и создастся дополнительный ток через резистор R2, в результате загорится нужный светодиод. Средний ток, протекающий через этот светодиод в режиме циклического опроса, должен быть достаточным для нормального свечения.

5.13. Способ идентификации нажатых клавиш [50]

При подаче на вход одновибратора F единичного сигнала он выдает импульс, длительность которого зависит от параметров внешней RC-цепи (рис. 5.36). В данном случае длительность импульса зависит от позиции нажатой клавиши. Опрос состояния клавиатуры производится так. МикроЭВМ выставляет в разряде 0 выходного порта лог. 1 и переходит к опросу разряда 7 входного порта, под-

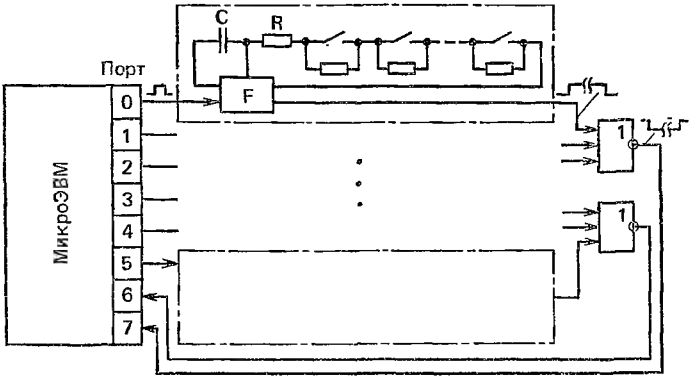


Рис. 5.36. Схема связи микроЭВМ с клавиатурой

считывая число циклов ожидания. При получении ответного сигнала (перехода из 0 в 1) по линии 7 опрос прекращается. По полученному числу циклов определяется ситуация на клавиатуре. Другой вариант: работа по прерыванию фоновой программы ответным сигналом от одновибратора со считыванием показаний внутреннего таймера.

5.14. Связь микроЭВМ с удаленным активным датчиком [51]

На рис. 5.37 показана схема сопряжения микроЭВМ с некоторым удаленным датчиком — в данном случае с клавиатурой. Сигналы с клавиатуры преобразуются в блоке сопряжения в последовательный код NRZ и передаются по магистрали M, содержащей две витые пары проводов: одну для передачи даиных D, другую для передачи синхроимпульсов C. Данные принимаются на сдвиговый регистр RG по положительным фроитам сигналов C и преобразуются в после-

Рис. 5.37. Схема связи микроЭВМ с удаленным активным датчиком

довательный код, поступающий на информационные входы q порта 2 микроЭВМ. Готовность информации в регистре подтверждается сигналом лог. 1 на входе Z прерывания микроЭВМ. В магистрали M использованы элемеиты D1 — D4 с открытым коллектором. В исходном состоянии x=1, y=1, регистр установлен в 0, элементы D1 — D4 выключены, на линиях магистрали поддерживаются высокие уровни напряжения.

После нажатия на клавишу блок сопряження анализирует состояние линии D. Наличие на линии D лог. 1 (напряжение высокого уровня) означает готовность регистра к приему данных. Если на линии присутствует лог. 0, то это означает, что микроЭВМ еще не приняла предыдущую посылку (8 бит). В этом случае блок сопряжения ждет появления сигнала готовности. Если готовность есть, то блок сопряжения анализирует состояние линии C. Условие C=0 в этом режиме означает, что сработал элемент D3, предназначенный для передачи из микроЭВМ сигнала общего сброса. По этому сигналу блок сопряжения прекращает работу и переходит в пассивное состояние.

При C=1 и D=1 блок сопряжения формирует стартовый бит (лог. 1) и серию из восьми информационных бит. Истинность передаваемых бит подтверждается положительными фронтами сигналов C. После заполнения сдвигового регистра стартовый бит достигает его верхиего выхода и в микроЭВМ посылается сигнал прерывания. В то же время блок сопряжения переходит в пассивное состояние, а в линии D благодаря элементу D4 поддерживается лог. 0. Получив сигнал прерывания, микроЭВМ считывает содержимое регистра и затем сбрасывает этот регистр сигналом y=0, после чего система возвращается в исходное состояние.

5.15. Управление отдельными битами выходного порта с использованием адресной шины [103]

Схема, показанная на рис. 5.38, позволяет одной комаидой произвести либо установку в 1 выбранного разряда выходного порта, либо сброс этого разряда, либо получить импульс на выходе этого разряда. В центральном процессоре типа 8085 предусмотреиа команда обращения к устройствам ввода-вывода. При ее выполнении сигнал $IO/\overline{M}=1$. Сигнал $\overline{W}/R=0$ при записи, $\overline{W}/R=1$ при чтении. При выполнении команды ввода-вывода в адресном байте A8 — A15 указывается адрес устройства ввода-вывода.

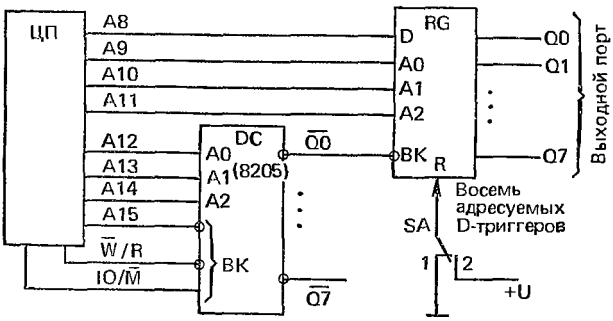


Рис. 5.38. Схема управления выходным портом

В данном случае разряд A8 предназначен для передачи данных (а не адресного бита), разряд A15 используется для обеспечения необходимых условий выбора кристалла ВК (см. гл. 1). Остальные 6 разрядов A9 — A14 позволят адресовать одну из $2^6=64$ линий выходного порта (показаны только 8 линий).

При выполнении команды Запись в устройство ввода-вывода, когда $A12=A13=A14=A15=0$, срабатывает дешифратор DC и на его выходе $\overline{Q0}$ формируется лог. 0. При этом в один из восьми триггеров регистра типа 4099 запишется значение сигнала (0 или 1), передаваемого из ЦП по линии A8. Код на линиях A9 — A11 определяет иомер одного из восьми триггеров выходного порта. Если перевести переключатель SA в положенне 2, то на выходной порт будет действовать сигиал сброса и записываемая единица будет держаться на его выходе только в течение времени, когда сигнал $\overline{W}/R$ имеет нулевое значение. В результате на выходе появится импульс (при тактовой частоте 6,14 МГц длительностью 0,6 мкс), хотя для обращения к выбранной линии порта была использована одна (а не две: 0→1, 1→0) команда вывода.

5.16. Периодическое отключение напряжения питания от энергонезависимых блоков микроЭВМ при выполнении циклических программ [96]

Схема части микроЭВМ, обеспечивающей управление напряжением питания, приведена на рис. 5.39. Процессор работает в циклическом режиме. Запуск очередного цикла производится при получении импульса с генератора импульсов. Предположим, что время между двумя соседними импульсами, поступающими с генератора, равно

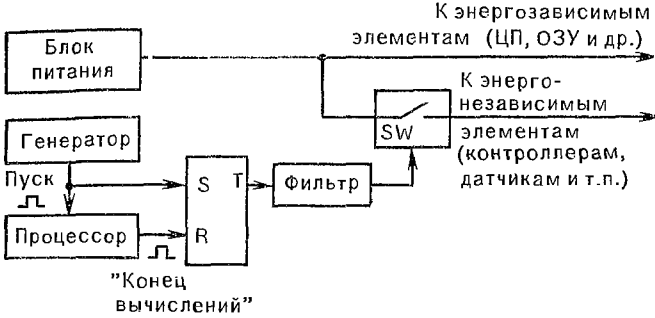


Рис. 5.39. Схема управления питанием блоков микроЭВМ

50 мс, а максимальное время выполнения программного цикла равно 12,5 мс. В этом случае в течение, как минимум, 37,5 мс микроЭВМ «простаивает» в каждом цикле. Для уменьшения потребляемой в этот период мощности производится размыкание ключа SW, выполненного, например, на транзисторе. При этом обесточиваются те элементы микроЭВМ, которым «нечего терять» при отключении питания. Остальные элементы, для которых отключение питания недопустимо, подключены непосредственно к блоку питания. Программно-доступный по входу R триггер устанавливается в единичное состояние, соответствующее замыканию ключа, в начале каждого цикла и устанавливается в 0, размыкая ключ, после окончания вычислений. Низкочастотный фильтр, например RC-цепь, служит для плавного включения-выключения ключа с целью предотвращения «ударов» по цепям питания.

5.17. Связь однокристальной микроЭВМ с линейным приемопередатчиком, клавиатурой и цифровым индикатором [1]

Рассмотрим пример, показывающий, как включить простейшую микроЭВМ в достаточно развитую систему (рис. 5.40). МикроЭВМ имеет два выходных порта R и Q и один входной порт K. Для подсветки цифр индикатора микроЭВМ поочередно, с достаточно высо-

кой частотой выдает «бегущую единицу» на выходы R2 — R12. Эти же сигналы используются для построчного опроса клавиатуриой матрицы. Импульсы на выходах R создаются программным путем (порт R представляет собой программно-доступный регистр с возможностью индивидуального управления выбранным битом); выходы Q микроЭВМ соединены с сегментиыми входами индикатора и с входами универсального асинхронного приемопередатчика.

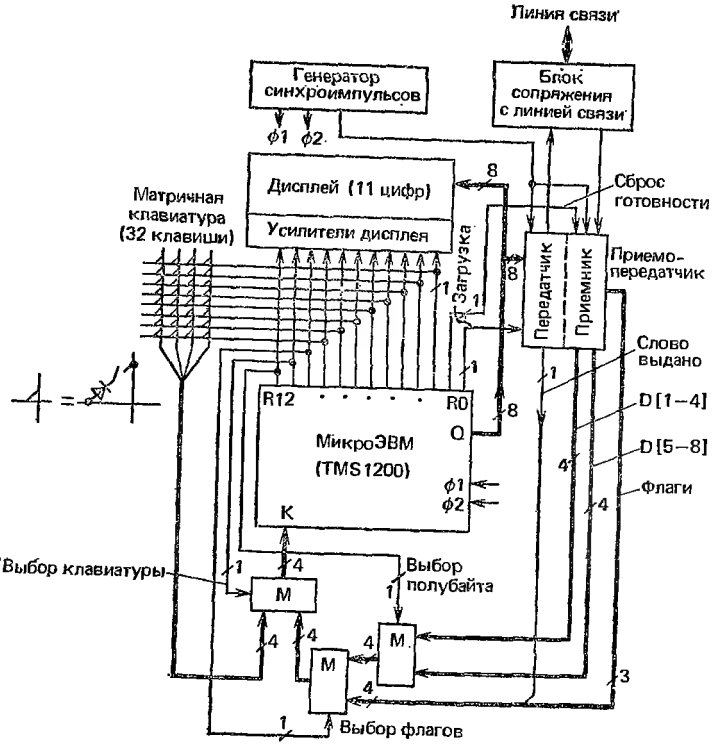


Рис. 5.40. Пример использования однокристальной микроЭВМ в терминальном устройстве

Если микроЭВМ «желает» передать в линию связи байт с выходов Q, то она программным путем создает импульс на выходе R0. Передающая секция приемопередатчика по этому сигналу принимает байт и выдает его в линию последовательным кодом. После передачи слова в линию передающая секция приемопередатчика формирует сигнал Слово выдано, который при наличии разрешения на выходе R10 передается через два последовательно включенных мультиплексора M (8×4 бит, типа SN74157) на один из входов K.

Приемная секция приемопередатчика способна принимать с линии связи последовательный код и «разворачивать» его в два параллельных полубайта. После того как эта операция выполнена, выставляются «флаги», характеризующие состояние приемника (Готов, Ошибка и др.). МикроЭВМ периодически «интересуется» состоянием флагов, пропуская их на входы К через два последовательно включенных мультиплексора. Для приема данных с приемной секции приемопередатчика микроЭВМ пропускает на входы К сначала один полубайт, а затем другой. Для передачи информации со столбцов клавиатурной матрицы открывается левый канал мультиплексора, подключенного непосредственно к входам К.

Управление входными потоками информации, таким образом, производится с помощью мультиплексоров в зависимости от комбинаций сигналов на выходах R, а они, в свою очередь, задаются программно. Все программы хранятся во внутреннем ПЗУ.

5.18. Пример использования однокристальной микроЭВМ в таксометре автомобиля [1]

Помимо накопления текущей стоимости проезда, вычисленной согласно некоторому тарифу, таксометр должен также вычислять стоимость проезда при изменяющейся расценке. Он должен обладать способностью умножать тариф на некоторый коэффициент, определяемый временем суток и днем недели (наибольшие расценки — в вечернее время выходных дней), делить стоимость проезда при наличии нескольких пассажиров-попутчиков и т. д. В данном примере с целью его упрощения рассматриваются простейшие функции.

Предполагается, что таксометр накапливает общую стоимость проезда, основываясь как на минимальной стоимости проезда, так и на тарифе, который вступает в действие после того, как пройден путь, при котором эта минимальная стоимость действовала. (Пассажиру тариф объявляется до того, как он вступает в действие.) Например, если минимальная плата за первую милю составляет 0,75 дол. и тариф при дальнейшем проезде 0,15 дол. за каждые пройденные 0,25 мили, то вначале фиксируется стоимость проезда 0,75 дол. до тех пор, пока такси не пройдет одну милю. В момент перехода на вторую милю к этой сумме прибавляется 0,15 дол. Когда такси пройдет 1,25 мили, к этой сумме вновь прибавляется 0,15 дол. и т. д. Система имеет стартовую кнопку, которая сбрасывает показания дисплея и инициирует новый цикл работы.

Система электронного таксометра показана на рис. 5.41. Для контроля пройденной дистанции используется индуктивный датчик, который выдает импульс на каждый оборот колеса (6,534 фута). Этот импульс усиливается и расширяется одновибратором. Для облегчения процедуры изменения констант при вычислениях используется диодная матрица. Можно было бы использовать для этой цели клавиатуру или наборные галетные диски, однако согласно стандартам США система, задающая тариф, должна быть защищена так, чтобы все изменения надежно контролировались.

Дистанция на диодной матрице представлена в виде числа оборотов колеса в двоичном коде. Например, если дистанция с минимальной оплатой составляет одну милю, или 808 оборотов колеса, то распайка диодов на диодной матрице должна соответствовать числу 328_{16} ($328_{16}=808_{10}$). Рассуждая аналогично, получаем, что

0,25 мили (оплачиваемый квант приращения дистанции) соответствует 202_{10} оборотам колеса, или, что то же самое, числу оборотов, равному $0CA_{16}$. Стоимость проезда задается на входах в виде двоично-кодированного десятичного числа (BCD-код). Например, минимальная стоимость 0,75 дол. представляется в виде числа 075.

Стартовая кнопка обладает «дребезгом», вызывающим серию стартовых импульсов при включении; программа должна обеспечивать компенсацию дребезга (чтобы не ставить триггерную схему защиты). Для отображения стоимости проезда используются пять 7-сегментных индикаторов и буферные микросхемы.

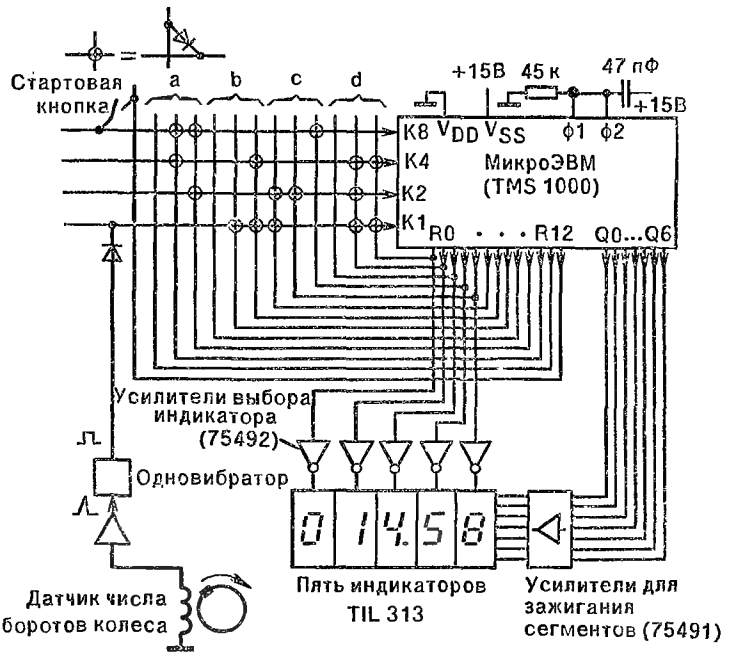


Рис. 5.41. Пример использования однокристальной микроЭВМ в таксометре:

a — оплачиваемый квант приращения дистанции, распаян код $0CA_{16}$; b — плата за приращение дистанции, распаян код 015_{10}; c — дистанция, соответствующая минимальной плате, распаян код 328_{16}; d — минимальная плата (при посадке в такси), распаян код 075_{10}

В процессе работы производится периодическая регенерация цифр дисплея и с помощью тех же (R0 — R4) и других (R5 — R12) выходов порта R сканирование множества 4-разрядных чисел, заданных с помощью диодов, а также опрос стартовой кнопки. Информация на выходах порта Q служит для подсветки нужных сегментов отображаемой цифры. При R0=R1=...=R11=0 восприни-

мается импульс с датчика; после окончания этого импульса можно переходить к опросу чисел, заданных на диодной матрице. Рабочая программа, хранимая во внутреннем ПЗУ, выполняется циклически, слежение за внешними объектами производится программно. Режим работы по прерыванию в данной микроЭВМ не предусмотрен.

5.19. Использование выходного порта микроЭВМ для ввода информации [115]

На рис. 5.42 показан выходной порт микроЭВМ (*а*) и временны́е диаграммы (*б*) сигналов на его линиях. Прямое назначение порта — передача данных к печатающему устройству.

Данные передаются байтами (D0 — D7), истинность каждого байта подтверждается импульсным сигналом STB (строб). При получении байта печатающее устройство временно снимает сигнал го-

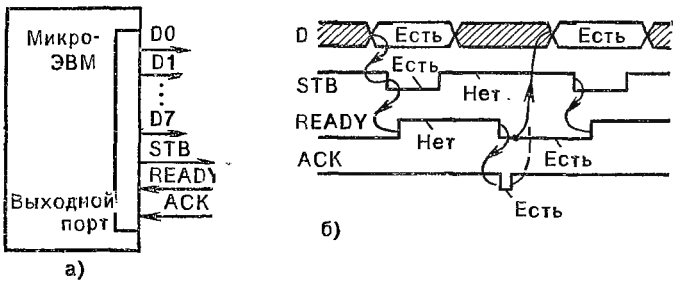


Рис. 5.42. Выходной порт микроЭВМ (*а*) и временны́е диаграммы его работы (*б*).

Данные D сопровождаются сигналом STB; READY, ACK — *ответные сигналы* от печатающего устройства

товности (READY=H). Когда печатающее устройство вновь оказывается готовым к приему очередного байта, оно формирует сигнал READY=L и вслед за этим — импульсный сигнал ответа ACK, который может быть подан на вход прерываний микроЭВМ.

Разряды D0 — D7 выходного порта доступны процессору по записи независимо от состояния ответных сигналов READY и ACK. Сигнал STB формируется автоматически и сопровождает данные при их выдаче из выходного порта. Сигнал READY может быть считан программно, при обращении процессора по определенному адресу. Сигнал ACK вызывает прерывание процессора по заданному вектору, если это не запрещено соответствующим разрядом регистра состояния порта.

В зависимости от принятого режима работы (асинхроиного или с прерыванием программы, см. § 2.3) процессор после выдачи в выходной порт очередного байта данных либо приступает к периодической программной проверке состояния сигнала READY, дожидаясь готовности печатающего устройства к приему следующего байта,

либо выполняет фоновую программу, разрешая прерывание по сигналу ACK. В дальнейшем, получив сигнал ACK, процессор временно откладывает выполнение фоновой программы и переходит к выполнению прерывающей программы, которая принимает к сведению факт готовности печатающего устройства и при необходимости выдает в выходной порт очередной байт данных.

Предлагается использовать выходной порт микроЭВМ не по прямому назначению — для обмена информацией с управляемым объектом. В качестве примера рассмотрим схему, показанную на рис. 5.43. Схема предназначена для управления электродвигателем ЭД и опроса состояния концевых датчиков К1 и К2 положения груза.

Рис. 5.43. Использование выходного порта микроЭВМ для управления электродвигателем ЭД и опроса состояния концевых датчиков К1, К2 положения груза

Если груз находится в крайнем нижнем положении, то замкнуты контакты датчика К1 и сигнал M1=1 (M2=0). Если груз находится в крайнем верхнем положении, то замкнуты контакты датчика К2 и сигнал M2=1 (M1=0). Если груз находится в промежуточном положении, то M1=M2=0.

Для перемещения груза вверх необходимо подать на входы схемы управления электродвигателем (Упр) комбинацию сигналов C0=1, C1=0. Для перемещения груза вниз нужна обратная комбинация этих сигналов: C0=0, C1=1. При C0=C1=0 электродвигатель остановлен. В исходном состоянии C0=C1=0.

Предположим, что программа пользователя предусматривает операцию включения электродвигателя для подъема груза. При выполнении этой операции микроЭВМ обращается к внешней системе, рассматривая ее как обычное печатающее устройство (см. рис 5.42, *б*). В разряды D0 и D1 выходного порта выдается нужная комбинация сигналов: D0=1, D1=0, а затем схемно вырабатывается импульс STB, по которому эти сигналы запоминаются в регистре RG (C0=1, C1=0). Единичный сигнал C0 поступает на элемент И — ИЛИ — НЕ и опрашивает состояние датчика К2.

Если груз находится в крайнем нижнем или промежуточном положении, то M2=0 и сигнал готовности отсутствует (READY=H),

Начинается подъем груза до тех пор, пока не замкнутся контакты датчика К2. При этом срабатывает элемент И — ИЛИ — НЕ и формируется сигнал READY=L. По фронту этого сигнала срабатывает одновибратор F и на его выходе формируется отрицательный импульс АСК (см. рис. 5.42, *б*). МикроЭВМ принимает к сведению, что операция выполнена и выключает электродвигатель, записав в регистр RG нулевой код.

Если при выдаче команды подъема груза оказалось, что груз уже находится в крайнем верхнем положении, то ответные сигналы READY и АСК поступают в микроЭВМ сразу же после выдачи команды, так как М2=1. МикроЭВМ, получив ответные сигналы, принимает к сведению, что команда выполнена и записывает в регистр RG нулевой код. Благодаря инерционности схемы Упр (содержащей

Рис. 5.44. Использование выходного порта микроЭВМ для считывания сигналов от источника информации

магнитные пускатели и другие элементы электроавтоматики), она «не замечает» кратковременный единичный сигнал С0, так что электродвигатель остается выключенным.

Выполнение команды опускания груза производится аналогично, с использованием кодовой комбинации С0=0, С1=1.

На рис. 5.44 показан пример использования выходного порта микроЭВМ для опроса восьми источников информации. В исходном состоянии в регистре RG записан нулевой код. Для считывания одного из входных сигналов X_i в регистр RG загружается код, содержащий единицу в выбранном разряде D_i и нули в остальных разрядах. Если выбранный входной сигнал $X_i=1$, то формируются ответные сигналы READY и АСК; если входной сигнал $X_i=0$, то ответных сигналов нет. Наличие ответных сигналов может проверяться процессором микроЭВМ сразу после загрузки регистра RG кодом, содержащим единицу в нужном разряде.

Можно воспользоваться режимом прерывания, если использовать сигнал АСК. Однако следует учесть, что этот сигнал формируется по фронту сигнала READY, а фронта может и не быть, если, например, все входные сигналы $X_i=1$, а перебор сигналов опроса ведется непрерывно, циклически, так что на выходе элемента И — ИЛИ — НЕ

присутствует статический сигнал лог. 0. Чтобы исключить такие ситуации, следует чередовать опрос с паузами, при которых после каждого «полезного» кода в регистр RG записывается нулевой код. Это позволяет при наличии всех единичных сигналов X_i получать динамический сигнал READY.

При работе в режиме прерывания микроЭВМ работает по фоновой программе до тех пор, пока выбранный сигнал X_i не примет значение лог. 1.

Если код в регистре RG содержит несколько единиц, то соответствующие входные сигналы X_i логически складываются и в микроЭВМ поступает обобщенный сигнал, характеризующий состояние выбранной группы входов.

При замене элемента И — ИЛИ — НЕ пирамидальной схемой из мультиплексоров можно использовать все $2^8=256$ возможных кодовых комбинаций в регистре RG для выбора одного из 256 источников информации. В режиме прерываний необходимо, как и ранее, чередовать опрос с паузами. Для этого, например, один из 256 информационных входов подключается к шине нулевого потенциала (лог. 0). Этот вход опрашивается в промежутках между опросами остальных входов. Следует также учесть, что при переключении комбинационной схемы из мультиплексоров на ее выходе могут появляться кратковременные импульсы, которые можно подавить с помощью, например, интегрирующей RC-цепи.

5.20. Система адресации удаленных устройств, использующая псевдослучайные числовые последовательности [116]

На практике часто возникает задача периодического опроса удаленных устройств с использованием последовательного канала связи.

На рис. 5.45, *а* показан пример решения этой задачи с использованием магистрали, в которой по отдельным линиям (витым парам проводов) передаются адрес, данные и синхросигналы. Число линий может быть уменьшено, например, до одной, с использованием кодов, описанных в § 3.1, однако рассмотренные далее недостатки системы, показанной на рис. 5.45, *а*, устранить таким путем не удается.

Основными недостатками системы являются ее низкая пропускная способность и отсутствие возможности обнаружения ложных сигналов в линиях A адреса и C синхронизации. Низкая пропускная способность связана с тем, что «магистральное время» используется в основном для передачи последовательного кода адреса выбранного контроллера K_i по линии A; при этом загрузка линии D данных невелика, особенно в тех системах, где от каждого контроллера при обращении к нему поступает только один бит данных (см. рис. 5.45, *б*). Ложные сигналы или пропуски правильных сигналов в линиях A и C могут привести к ложным опознаниям адреса контроллерами K_i и, следовательно, к поступлению в процессор неправильных данных.

Чтобы повысить пропускную способность системы и обеспечить повышенную помехоустойчивость по каналам A и C, предлагается использовать для адресации контроллеров K_i (рис. 5.45, *а*) генератор псевдослучайной последовательности чисел, который может быть выполнен аппаратным или программным, в зависимости от требова-

ний, предъявляемых к скорости опроса контроллеров $К_i$. Генератор псевдослучайных чисел вырабатывает непрерывную циклическую последовательность бит, условно показанную на рис. 5.46, *а* в виде кольца. При работе системы фронт сигнала в линии С подтверждает истинность бита, установленного в линии А.

Псевдослучайная последовательность адресных бит анализируется контроллерами $К_i$. Каждый контроллер ищет в проходящем по

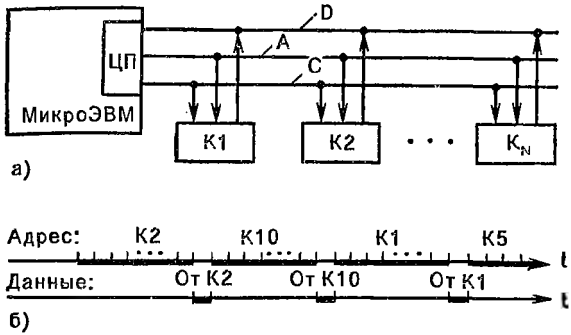


Рис. 5.45. Традиционная система *(а)* опроса удаленных контроллеров $К_i$. Основная часть магистального времени *(б)* тратится на передачу последовательного кода адреса («вопросы длинные, а ответы короткие»)

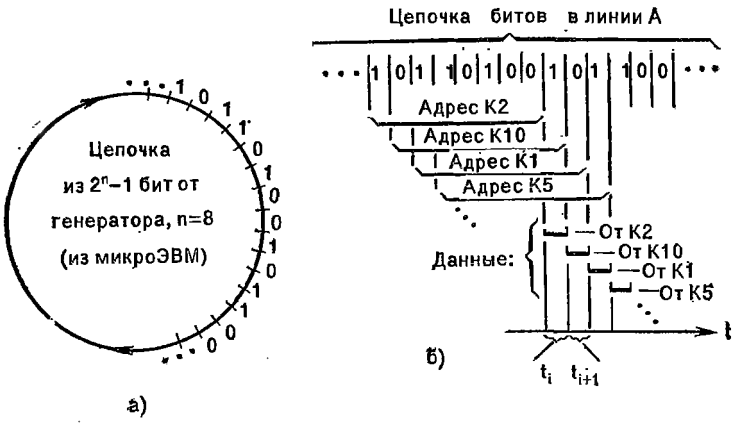


Рис. 5.46. Предлагаемая система опроса.

Процессор программно или аппаратно генерирует непрерывную последовательность (*а*) бит, которая просматривается всеми контроллерами одновременно, так что сразу несколько контроллеров находятся на разных стадиях дешифрации собственного адреса (*б*); при этом обеспечивается непрерывность потока данных по линии D

линии А потоке бит код, совпадающий с собственным адресом. В примере, приведенном на рис. 5.46, *б*, собственный адрес контроллера К2 равен 10110100, собственный адрес контроллера К10 равен 01101001, собственный адрес контроллера К1 равен 11010010 и т. д.

Из рис. 5.46, *б* следует, что собственные адреса контроллеров К2, К10, К1 частично «перекрываются» и в то же время эти адреса отличны друг от друга. Продолжая процесс назначения собственных адресов контроллеров $К_i$ просмотром псевдослучайной последовательности адресных бит через движущееся вправо 8-разрядное «окно», получаем очередной собственный адрес 10100101, который в данном примере присваивается контроллеру К5. Этот процесс

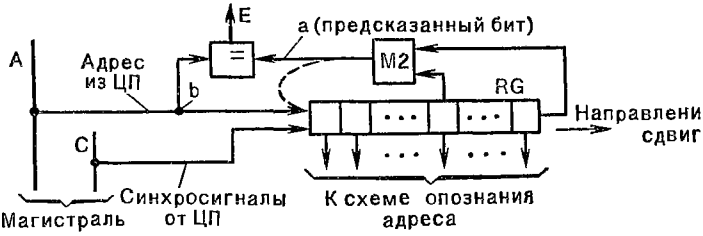


Рис. 5.47. Схема самосинхронизации системы, входящая в состав каждого контроллера $К_i$.

Основной элемент схемы — генератор заданной псевдослучайной последовательности с разомкнутой обратной связью

в конечном счете приводит к назначению 255 (2^n—1, n=8) различных собственных адресов контроллерам К1 — К255 (нулевой собственный адрес не используется).

После опознания адреса контроллером К2, в интервале времени t_i, рис. 5.46, *б*, этот контроллер посылает в линию D бит данных — 0 или 1. Этот бит принимается процессором. В интервале времени t_i контроллер К10 принимает из линии 4 последний бит адресного кода, который совпал с его собственным адресом. Поэтому в интервале времени t_{i+1} контроллер К10 посылает в линию D бит данных, а в последующие интервалы времени линия D используется для передачи данных от контроллеров К1, К5 и т. д. Сравнивая временные диаграммы, приведенные на рис. 5.45, *б* и 5.46, *б*, можно заключить, что использование предлагаемого метода адресации позволяет максимально «уплотнить» поток данных по линии D, т. е. повысить пропускную способность системы.

Для обнаружения сбоев, например из-за помех в магистрали, в каждом контроллере содержится собственный генератор той же псевдослучайной последовательности, которая вырабатывается процессором (рис. 5.47). Особенность этого генератора, выполненного по известной схеме (см., например, [66]), состоит в том, что обратная связь, показанная штриховой линией, разомкнута. При правильной работе системы «предсказанный» бит в точке a должен совпадать с «фактическим» битом, поступившим из адресной линии магистрали в точку b. Если обнаружено несовпадение, то вырабатывается сигнал ошибки E, по которому контроллер временно отключается от

системы — до тех пор, пока ошибочный бит пройдет сквозь весь сдвиговый регистр RG и затем будет зарегистрировано подряд n правильных сравнений предсказанных и фактических значений адресных бит.

Выполнение этого условия означает, что достигнут полный синхронизм между генератором процессора и генератором контроллера и достоверность последовательности адресных бит в сдвиговом регистре контроллера гарантирована.

Отключенный в результате обнаружения ошибки контроллер не выдает в магистраль «полезную» информацию. Например, если «полезная» информация (лог. 0 и лог. 1) представлена в линии D токами противоположного направления, то отключенный контроллер не возбуждает тока вообще, что расценивается процессором как сообщение об отсутствии информации от адресуемого в данном интервале времени контроллера.

Описанный механизм самосинхронизации позволяет исключить какие-либо процедуры начальной установки системы после включения напряжения питания — все контроллеры сами приводятся в нужное состояние, в точности отвечающее состоянию центрального процессора.

Предусмотрен также режим передачи информации из центрального процессора в контроллеры, который реализуется следующим образом.

1. В исходном состоянии процессор ведет циклический опрос контроллеров, как было описано ранее; в контроллеры передается псевдослучайная последовательность адресных сигналов, а из контроллеров — последовательность бит данных.

2. Процессор, желая передать один или несколько бит данных в контроллер с адресом A*, дожидается момента окончания выдачи этого адреса в магистраль в процессе циклического опроса контроллеров (см. п. 1).

3. После передачи последнего бита адреса A* процессор выдает в линию A неправильный бит, нарушающий заранее известную псевдослучайную последовательность адресных сигналов.

4. Благодаря наличию в каждом контроллере эталонного генератора заданной псевдослучайной последовательности (рис. 5.47), ошибка, преднамеренно внесенная процессором в эту последовательность, регистрируется всеми контроллерами. В результате выдача данных из контроллеров в магистраль прекращается, все контроллеры переходят в пассивное состояние на время продвижения ошибочного бита через сдвиговый регистр RG (рис. 5.47) и последующего выполнения процедуры самосинхронизации, как было описано ранее. Таким образом, процессор добивается освобождения линии данных на относительно большой промежуток времени, что позволяет ему в дальнейшем воспользоваться этой линией для передачи одного или нескольких бит данных в контроллер с адресом A*.

5. После передачи ошибочного бита процессор продолжает формировать заданную псевдослучайную последовательность сигналов, начиная с повторной выдачи всех битов адреса A*. Иными словами, процессор возвращается к моменту начала передачи адреса A* и следующих за ним адресов, но на этот раз шина данных не занята и процессор может ею воспользоваться.

6. Процессор последовательно передает в адресную линию все биты кода A*.

7. Контроллер, имеющий адрес A*, повторно опознаёт свой

адрес в линии A и расценивает этот факт (двойное обращение, разделенное ошибочным битом) как команду приема данных от процессора. Остальные контроллеры ни до, ни после получения неправильного адресного бита не опознают свой адрес и рассматривают возникшую ситуацию как «обычный» сбой, т. е. временно переходят в пассивное состояние и затем, войдя в синхронизм с процессором, возобновляют работу.

8. После выдачи последнего разряда кода A* в линию A процессор передает в линию D один или несколько бит данных, которые принимаются контроллером, опознавшим свой адрес.

9. Выбранный контроллер принимает данные и переходит в исходное состояние ожидания следующего обращения. Остальные контроллеры также переходят в это состояние после восстановления правильной синхронизации, как было описано ранее.

Таким образом, при передаче данных от процессора к контроллеру в последовательность адресных сигналов преднамеренно вводится ошибка. Если же произошла «настоящая» ошибка, то она не сопровождается повторением только что переданного адресного кода, во всяком случае, вероятность именно такого уникального сочетания ложных сигналов пренебрежимо мала.

5.21. Упрощение конструкции системы электрического согласования кабеля с устройствами — источниками и приемниками информации [117, 208]

На рис. 5.48, *а* показана схема подключения к порту микроЭВМ четырех блоков B_i сопряжения с накопителями D_i на гибких магнитных дисках. Подключение производится с помощью плоского (ленточного) кабеля с Т-образными ответвителями для связи с блоками B1, B2, B3. В кабеле имеются общие для всех блоков B_i линии передачи группы управляющих и информационных сигналов F (рис. 5.48, *б*) и индивидуальные (радиальные) линии для передачи из микроЭВМ сигналов SEL_i выбора блока B_i. Каждая радиальная линия соединяет определенный выходной разряд порта микроЭВМ с верхним входом одного из контроллеров. Эта линия продолжения в кабеле не имеет. Резистор R′ служит для согласования с волновым сопротивлением линии связи.

Задача состоит в том, чтобы обеспечить электрически правильную передачу по кабелю сигналов F, общих для всех блоков B_i. К этим сигналам относятся: Включение электродвигателя, Шаг (с дорожки на дорожку), Направление (перемещения головки), Данные и др. Однонаправленные линии связи для передачи сигналов от микроЭВМ должны быть электрически согласованы со стороны наиболее удаленного приемника (в блоке B4), двунаправленные линии согласуются с обеих сторон, см. гл. 4.

Общепринятое решение этой задачи заключается в установке в блоке B_i разъема, в который при необходимости может устанавливаться колодка с набором резисторов R. На рис. 5.48, *а* показано, что колодка с набором резисторов установлена только в наиболее удаленном от микроЭВМ блоке B4, а в остальных блоках разъемы остаются незанятыми.

Недостаток такого решения связан именно с наличием разъемов. Каждый разъем должен быть физически доступен пользователю, причем должно быть достаточно места, чтобы вручную, без специаль-

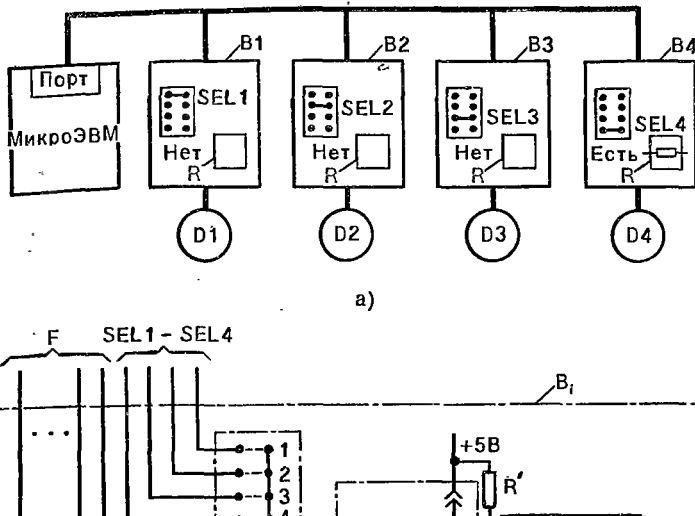
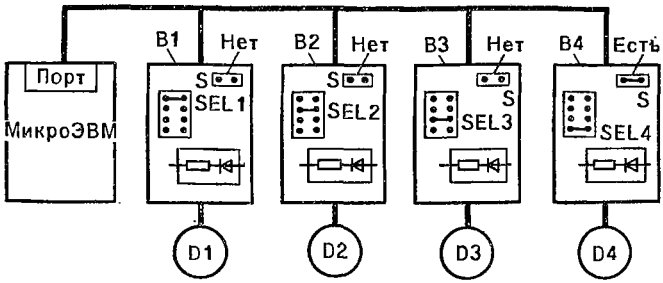


Рис. 5.48. Система связи микроЭВМ с накопителями на гибких магнитных дисках *(а)* и структура блока сопряжения *(б)*.

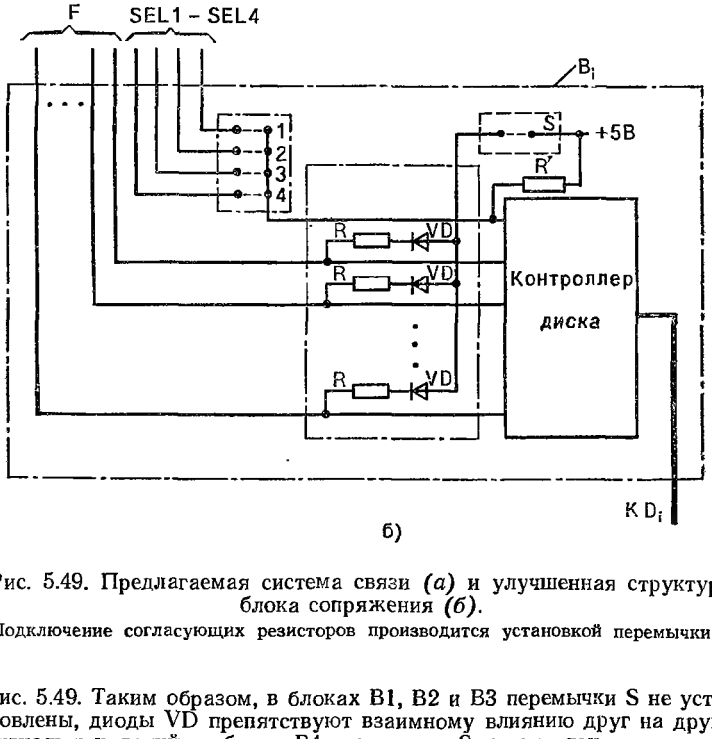
В каждом блоке имеется разъем для подключения колодки с согласующими резисторами

ных приспособлений устанавливать и извлекать набор резисторов при изменении конфигурации системы. Так как блок B_i обычно конструктивно совмещен с накопителем D_i, указанный недостаток препятствует уменьшению габаритов системы ввода-вывода на гибких магнитных дисках.

Предлагается исключить этот разъем из блока B_i, согласующие резисторы R устанавливать всегда, но отключать их не «физически», а электрически снятием напряжения +5 В с анодов развязывающих диодов VD при размыкании перемычки (или переключателя) S,



а)



б)

Рис. 5.49. Предлагаемая система связи *(а)* и улучшенная структура блока сопряжения *(б)*.

Подключение согласующих резисторов производится установкой перемычки S

рис. 5.49. Таким образом, в блоках B1, B2 и B3 перемычки S не установлены, диоды VD препятствуют взаимному влиянию друг на друга сигнальных линий; в блоке B4 перемычка S установлена и резисторы R, как обычно, выполняют функции согласования с волновым сопротивлением кабеля.

Предложенную схему можно, по-видимому, усовершенствовать и далее, если, например, ввести не ручное, а автоматическое замыкание — размыкание переключателя S в зависимости от наличия «соседа справа». Если таковой имеется, то переключатель S должен быть разомкнут, если нет — замкнут (рис. 5.50). В этой схеме, од-

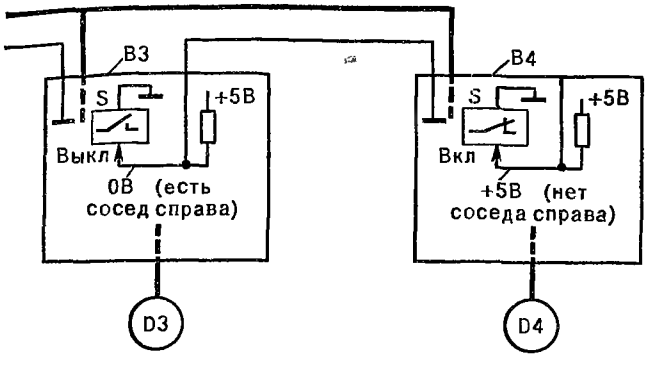


Рис. 5.50. Автоматическое подключение согласующих резисторов к физическому концу кабельной линии связи

нако, появляется дополнительный «ныряющий» в блоки B_i провод, т. е. усложняется конструкция соединительного кабеля. Но при этом исключается необходимость доступа со стороны оператора к переключателям S, устраняется возможность ошибки оператора при установке этих переключателей в нужные положения.

5.22. Использование сетевого таймера для синхронизации данных, поступающих в микроЭВМ от удаленного источника информации [12]

Схема устройства для сопряжения источника информации с микроЭВМ, показанная на рис. 5.51, обеспечивает передачу данных по запросу от микроЭВМ. Скорость передачи — 50 бит/с, передаваемые биты синхронизируются частотой 50 Гц питающей сети переменного тока напряжением 220 В.

Устройство содержит блок 1 связи с источником информации и блок 2 связи с микроЭВМ, соединенные двухпроводной линией связи 3. Блок 1 содержит блок питания 4, логический блок 5 и формирователь импульса 6. Выход положительного напряжения блока 4 соединен с шиной 7 положительного напряжения блока 5 и с входом питания формирователя 6. Выход нулевого потенциала блока 4 соединен с шиной 8 нулевого потенциала блока 5 и источника информации. Входы формирователя 6 соединены с выходами 9 переменного напряжения блока 4. Выход формирователя 6 соединен с входом 10 блока 5.

Входы 11 блока 5 соединены с выходами источника информации (не показан). Шины 12 и 13 блока 2 предназначены для подключения отрицательного и положительного полюсов источника питания линии связи 3; шины 14 и 15 этого блока являются шиной нулевого потенциала микроЭВМ и шиной питания блока 2. Вход 16 и выход 17 блока 2 соединены с выходом опроса и с информационным входом микроЭВМ. Провода 18 и 19 сети переменного тока (220 В,

50 Гц) подключены к входам блока 4 и к входам блока питания микроЭВМ (не показан).

Блок питания 4 (рис. 5.52) содержит силовой трансформатор 20, двухполупериодный выпрямитель 21, фильтрующий конденсатор 22 и стабилизатор 23. Формирователь импульса 6 (рис. 5.53) содержит диоды 24, 25, резисторный делитель напряжения 26—27 и триггер Шмидта 28. Логический блок 5 (рис. 5.54) содержит сдвиговый регистр 29, элементы И 30, И — НЕ 31, усилитель 32, формирователь импульса 33, триггер 34, резистор 35, передающий 36 и приемный 37 оптроны. Внешний информационный вход 38 сдвига регистра 29 соединен с шиной лог. 1, так что при сдвиге информации в направлении, указанном стрелкой, освободившиеся разряды заполняются единицами.

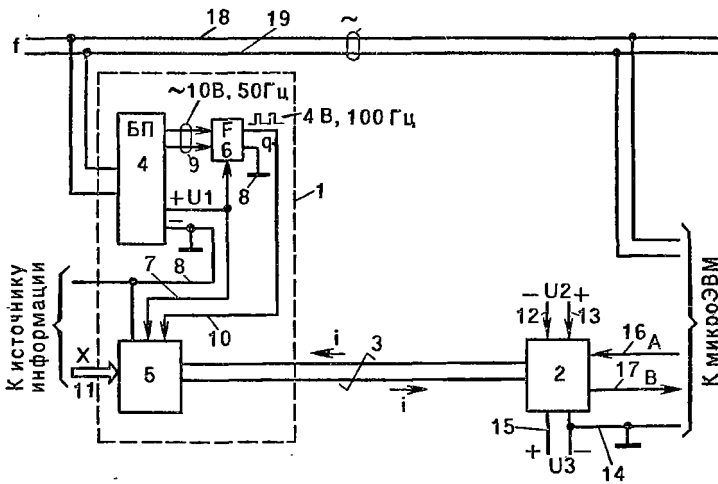


Рис. 5.51. Система связи микроЭВМ с удаленным источником информации

Блок 2 связи с микроЭВМ (рис. 5.55) содержит передающий оптрон 41, приемный оптрон 42, усилитель 43, резисторы 44 и 45.

Рассмотрим работу составных частей устройства.

Блок питания 4 (см. рис. 5.51, 5.52) предназначен для выработки стабилизированного постоянного напряжения U1 (5 В) питания блока 5 и формирователя 6, а также для формирования переменного напряжения 10 В, 50 Гц, подаваемого на выходы 9. Силовой трансформатор 20 снижает напряжение от 220 до 10 В и обеспечивает гальваническую развязку сигнальных цепей от сети переменного тока (провода 18, 19). Двухполупериодный выпрямитель 21 формирует пульсирующее напряжение частотой 100 Гц, которое частично (примерно на 80 %) «сглаживается» конденсатором 22 и поступает на вход стабилизатора 23, с выходов 7 и 8 которого снимается постоянное напряжение 5 В с нестабильностью около 1 %.

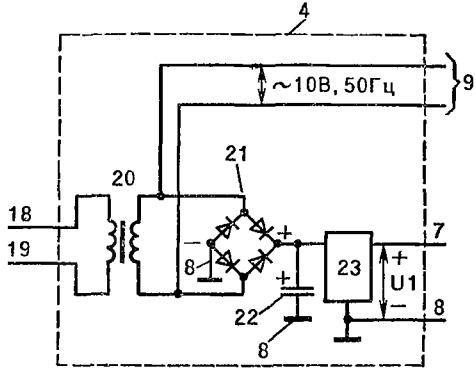


Рис. 5.52. Пример схемы блока питания

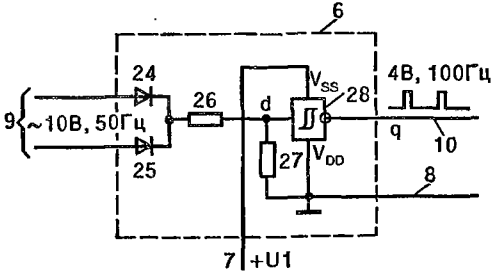


Рис. 5.53. Пример выполнения схемы формирователя импульса

Формирователь импульса 6 (рис. 5.53) вырабатывает последовательность положительных импульсов частотой 100 Гц. Переменное напряжение частотой 50 Гц, поступающее на входы 9, преобразуется диодами 24 и 25 в пульсирующее напряжение частотой 100 Гц, которое уменьшается по амплитуде до 4,5 В с помощью делителя напряжеиия на резисторах 26—27 и поступает на вход триггера Шмидта 28; с его выхода снимается непрерывная последовательность импульсов частотой 100 Гц.

Сдвиговый регистр 29 (рис. 5.54) работает в двух режимах: приема и сдвига. В режиме приема сигнал S=1 (напряжение высокого уровня), в режиме сдвига S=0. Прием информации с входов регистра или ее сдвиг производится в момеит поступления положительного перепада напряжения на вход синхронизации (сигнал CL). Формирователь импульса 33 вырабатывает положительные импульсы при переходе входного сигнала из состояния лог. 0 в состояние лог. 1.

Триггер 34 работает в режиме делителя частоты сигналов: входной сигнал q имеет частоту 100 Гц; выходной — частоту 50 Гц и скважность, равную двум. Сигнал лог. 0, поступающий на вход

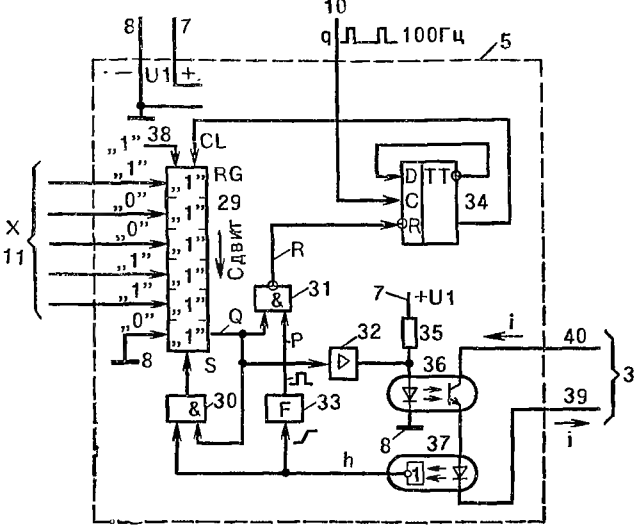


Рис. 5.54. Схема логического блока 5 (см. рис. 5.51)

установки нуля R триггера 34, в необходимых случаях обеспечивает фазовую синхронизацию работы микроЭВМ и сдвигового регистра 29.

Усилители 32, 43 (рис. 5.54, 5.55) выполнены по схеме с «открытым коллектором». Передающие оптроны 36, 41 включены (напряжение коллектор — эмиттер при этом составляет примерно 0,4 В) при излучении света соответствующими светодиодами и выключены при отсутствии света. Приемные оптроны 37, 42 формируют на информационных выходах сигналы лог. 0 при излучении света соответствующими светодиодами и сигналы лог. 1 при отсутствии света. Питание блока 5, линии связи 3 и блока 2 осуществляется соответственно от источников напряжения U1, U2 и U3, гальванически не связанных между собой для повышения помехоустойчивости.

В исходном состоянии (см. рис. 5.56, период времени до момента t_0) из микроЭВМ на вход 16 устройства поступает сигнал A=1 (отсутствие запроса на сеанс связи). Через светодиод оптрона 41 протекает ток, транзистор этого оптрона насыщен. На вход синхронизации регистра 29 поступает сигнал CL частотой 50 Гц. Этот регистр заполнен «единицами» (любое другое начальное состояние регистра оказывается неустойчивым) и находится в режиме сдвига, так как S=0. Усилитель 32 выключен по выходу, так как на его вход поступает сигнал Q=1. Поэтому через светодиод оптрона 36 протекает ток и транзистор этого оптрона насыщен.

В линии связи 3 протекает ток i, поэтому на выходах оптронов 37 и 42 сформированы сигналы h=0 и B=0. Сигнал h=0 проходит через элемент И 30 и подтверждает исходное состояние сигнала S=

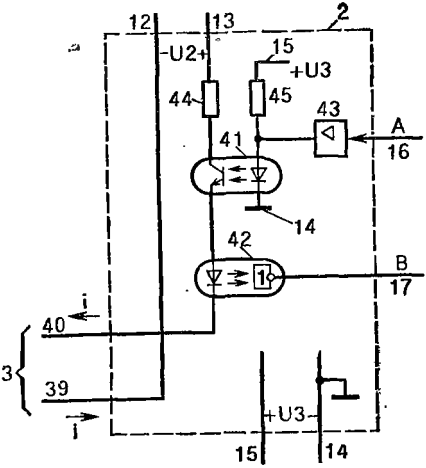


Рис. 5.55. Схема блока 2 связи с микроЭВМ (см. рис. 5.51)

=0. На выходе формирователя импульса 33 присутствует сигнал P=0, так как на его вход постоянно подан сигнал h=0. Элемент И—НЕ 31 закрыт сигналом P=0, поэтому на его выходе постоянно присутствует сигнал R=1, который не оказывает воздействия на триггер 34, работающий в режиме делевия частоты.

До момента t_0 микроЭВМ работает по программе, не связанной с опросом источника информации. В дальнейшем, в моменты t_0, t_2, t_4, t_6, ... и т. д. микроЭВМ совершает кратковременные обращения к прерывающей программе обслуживания устройства (эти обращения условно показаны кратковременными положительными импульсами на временно́й диаграмме 57, приведенной на рис. 5.56). В момент t_0 микроЭВМ начинает, а в момент t_2 завершает операцию считывания входного кода X с линий

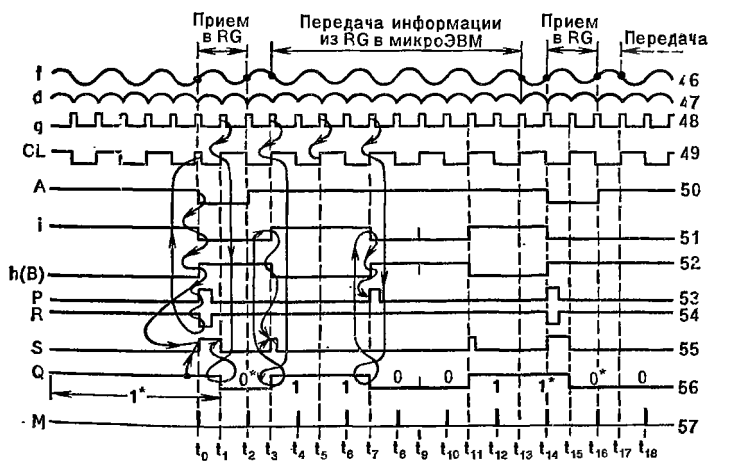


Рис. 5.56. Временна́я диаграмма работы системы связи (см. рис. 5.51)

11 в регистр 29. В моменты t_4, t_6, t_8, t_{10}, t_{12} микроЭВМ опрашивает состояние линии 17, на которой формируется последовательность бит, соответствующих принятому в регистр 29 коду X. Если B=0, то принят единичный бит, если B=1, то нулевой бит.

Прерывающая программа обслуживания устройства вызывается по сигналам от сетевого таймера (см. § 1.2), который входит в состав блока питания микроЭВМ. Таймер вырабатывает сигналы прерывания работы процессора микроЭВМ, причем эти сигналы формируются в моменты, на несколько десятков микросекунд предшествующие прохождению напряжения первичной питающей сети 220 В, 50 Гц через «нуль» в определенном направлении — на рис. 5.56 моменты прерывания (диаграмма 57) соответствуют переходу напряжения из «отрицательной» области в «положительную». Отметим, что моменты выдачи сигналов сетевым таймером в равной мере могут соответствовать переходу напряжения из «положительной» области в «отрицательную» — все определяется рядом случайных факторов: разводкой проводов силовой сети в помещении (к «правому» гнезду двухполюсной розетки с равной вероятностью может быть подключен первый или второй провод сети), включением двухполюсной вилки в розетку («правый» полюс вилки может с равной вероятностью попасть в «правое» или «левое» гнездо розетки), разводкой проводов, соединенных с первичной и вторичной обмотками силового трансформатора блока питания микроЭВМ и другими подобными факторами. Поэтому если задаться системой отсчета времени, связанной с односторонними переходами сетевого напряжения через «нуль», то заранее не известно, синфазны или противофазны эти переходы моментам перехода микроЭВМ к обработке прерываний от сетевого таймера.

Идея работы предлагаемого устройства состоит в обеспечении противофазной работы сдвигового регистра 29 и микроЭВМ. При противофазной работе передаваемые биты стабильны в те моменты времени, когда микроЭВМ их принимает, обращаясь к прерывающей программе по сигналам сетевого таймера. При этом смена информации производится в промежутках между сигналами прерывания, когда микроЭВМ работает по фоновой программе, не связанной с обслуживанием устройства (см. рис. 5.56, диаграммы 56 и 57).

В момент t_0 микроЭВМ временно прекращает выполнение фоновой программы, запоминает параметры возврата (содержимое счетчика команд и регистра состояния процессора, а также при необходимости другие данные) и переходит к выполнению прерывающей программы, связанной с обслуживанием устройства. При начальном вхождении в эту программу микроЭВМ выполняет команду выдачи в выходной порт сигнала A=0, после чего микроЭВМ выходит из режима прерывания, т. е. восстанавливает в регистрах ранее сохраненную информацию и вновь приступает к выполнению фоновой программы, начиная с прерванного места.

Сигнал A=0 вызывает выключение оптрона 41 и, следовательно, прекращение тока i в линии связи 3. Вследствие этого на выходе приемного оптрона 37 формируется сигнал лог. 1, который переводит регистр 29 в режим приема информации (S=1) и вызывает срабатывание формирователя импульса 33. На выходе элемента И — НЕ 31 формируется отрицательный (по фазе) импульс, который устанавливает триггер 34 в нуль. Начиная с этого момента, сдвиговый ре-

гистр 29 и микроЭВМ работают синхронно, с фазовым сдвигом 180°.

На рис. 5.56 показана «неблагоприятная» ситуация, когда до момента t_0 сигнал CL формировался не в той фазе, которая требуется. Из временных диаграмм видно, что после момента t_0 сигнал CL меняет фазу. Так как положительный фронт сигнала q сформирован раньше, чем положительный фронт сигнала h, исключены ситуации, связанные с изменением начального состояния разрядов регистра 29 (все разряды остаются в состоянии лог. 1).

В момент t_1 под действием положительного фронта сигнала CL в регистр 29 принимается код X и служебный сигнал лог. 0 (помечен на рис. 5.56 знаком *). Сигнал Q=0 запирает элемент И 30 и переводит регистр 29 в режим сдвига (S=0). Линия связи 3, уже разомкнутая оптроном 41, дополнительно размыкается оптроном 36, так как на выходе усилителя 32 сформирован сигнал лог. 0.

В момент t_2 микроЭВМ вновь обращается к прерывающей программе, которая на этот раз предписывает машине перевести сигнал A выходного порта в состояние лог. 1. При этом оптрон 41 включается, но ток i по-прежнему отсутствует, так как оптрон 36 остается выключенным. Далее микроЭВМ возвращается к выполнению фоновой программы.

В момент t_3 под действием положительного фронта сигнала CL информация в регистре 29 сдвигается на один разряд вниз; при этом на выходе Q формируется сигнал лог. 1 (см. код, приведенный в качестве примера на входах регистра 29, рис. 5.54), включается оптрон 36, в линии 3 начинает течь ток i, сигналы h и B принимают значения лог. 0. На входе режима работы регистра 29 формируется «ложный» импульс (S=1), который, однако, не приводит к возникновению «гонок», так как фронт импульса CL уже прошел.

В момент t_4 микроЭВМ, выполняя прерывающую программу, считывает из входного порта сигнал B=0, что соответствует лог. 1 в младшем разряде кода X. МикроЭВМ программно «упаковывает» полученный бит в свою память и возвращается к фоновой задаче.

В момент t_5 информация в регистре 29 сдвигается вниз на один разряд; сигнал Q в данном примере по-прежнему равен лог. 1.

В момент t_6 микроЭВМ принимает второй бит кода X.

В момент t_7 под действием положительного фронта сигнала CL информация в регистре 29 сдвигается вниз на один разряд; сигнал Q принимает нулевое значение. Оптрон 36 выключается и ток i в линии связи 3 прекращается. Сигнал h на выходе оптрона 37 принимает единичное значение, поэтому на выходе формирователя 33 вырабатывается импульс, который, однако, не поступает на вход R триггера 34 благодаря тому, что Q=0 и элемент И — НЕ 31 закрыт.

В момент t_8 микроЭВМ считывает третий бит кода X. В момент времени t_9 производится сдвиг кода, а в момент t_{10} — опрос четвертого бита. В момент t_{11} сигнал Q принимает единичное значение, на входе S формируется «ложный» импульс, не нарушающий правильной работы устройства (так же как и в момент t_3).

В момент t_{12} микроЭВМ принимает последний (пятый) бит кода X. По мере сдвига полезной информации регистр 29 заполняется единицами, поступающими с шины 38. В момент t_{13} первая из этих единиц (помечена на рис. 5.56 знаком *) достигает выхода Q и устройство переходит в описанное выше исходное состояние.

Разрядность кода X может быть произвольной, однако для наглядности рис. 5.56 принята разрядность, равная пяти.

В момент t_{14} микроЭВМ начинает новый сеанс связи. На этот раз, однако, сигнал CL имеет нужную фазу и сброс триггера 34 лишь подтверждает его текущее состояние.

После начальной установки устройства (моменты t_{14}, t_{16}) передается младший разряд нового кода X (момент t_{18}) и далее процесс протекает по уже описанному алгоритму.

5.23. Использование сетевого таймера для синхронизации обмена данными между двумя микроЭВМ [14]

Предлагаемое устройство (рис. 5.57) обеспечивает обмен информацией между двумя микроЭВМ, удаленными друг от друга на расстояние порядка 1 км. Обе микроЭВМ подключены к общей сети переменного тока напряжением 220 В. Частота переменного тока в сети (50 Гц) используется для синхронизации передаваемых бит благодаря использованию сетевых таймеров обеих микроЭВМ. Информация передается по запросу микроЭВМ-приемника. При приеме

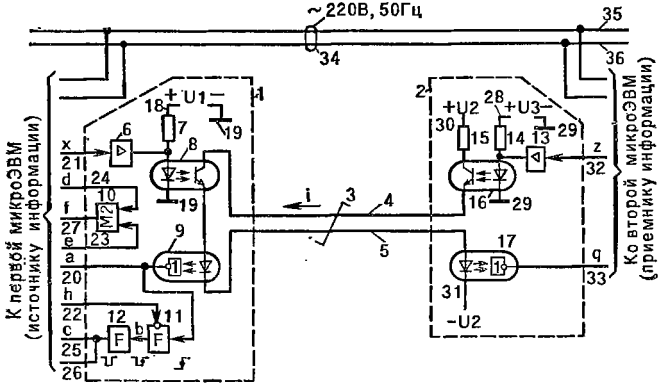


Рис. 5.57. Система связи между двумя микроЭВМ.

Переменное напряжение питающей сети 220 В, 50 Гц используется для синхронизации передаваемых бит

запроса производится противофазная синхронизация таймеров, так что моменты выдачи и приема бит чередуются во времени с интервалом 10 мс. Запросный и информационные сигналы передаются по одной и той же двухпроводной линии связи.

Устройство содержит блоки 1 и 2 связи с первой и второй микроЭВМ, соединенные линией связи 3, включающей провода 4 и 5.

Блок 1 содержит усилитель 6, резистор 7, передающий 8 и приемный 9 оптроны, элемент Исключающее ИЛИ 10, а также формирователи импульса 11 и 12. Блок 2 содержит усилитель 13, резисторы 14, 15, передающий 16 и приемный 17 оптроны. Выходы 26 и 27 блока 1 соединены с входами прерывания первой микроЭВМ. Выходы 20 и 25 блока 1 соединены с двумя разрядами входного порта первой микроЭВМ. Входы 21—24 блока 1 соединены с четырьмя раз-

рядами выходного порта первой микроЭВМ. Вход 32 и выход 33 блока 2 соединены соответственно с разрядом выходного и входного портов второй микроЭВМ. Питание первой и второй микроЭВМ осуществляется от общей сети 34 переменного тока напряжением 220 В частотой 50 Гц, включающей провода 35 и 36.

Каждая микроЭВМ содержит сетевой таймер (см. § 1.2), который вырабатывает логический сигнал, синфазный или противофазный сигналу переменного напряжения сети. Например, сигнал лог. 1 вырабатывается сетевым таймером при положительной полуволне напряжения между проводами 35 и 36, а сигнал лог. 0 — при отрицательной полуволне.

Так как обе микроЭВМ подключены к одной и той же сети переменного тока, таймеры этих микроЭВМ работают синхронно. Однако заранее неизвестно, работают ли оба таймера синфазно или противофазно относительно друг друга. Это связано с тем, что заранее неизвестно, как подведены провода 35 и 36 к розеткам в помещении («нуль» в правом или левом гнезде розетки), как вставлена двухполюсная вилка шнура питания микроЭВМ в розетку (с поворотом на 180° или без поворота), как разведены провода, связанные с силовым трансформатором блока питания микроЭВМ и т. д.

Для правильной работы системы из двух микроЭВМ необходимо, чтобы моменты приема информационных бит микроЭВМ-приемником и их смены микроЭВМ-источником были разнесены во времени. Это условие обеспечивается благодаря тому, что в начале сеанса связи, при приеме сигнала запроса на передачу информации, производится противофазная синхронизация таймеров, так что моменты выдачи и приема бит чередуются во времени, т. е. в момент приема очередного информационного сигнала с линии связи он заведомо стабилен (лог. 0 или лог. 1).

В исходном состоянии обе микроЭВМ работают по программам, не связанным с обменом информацией. Из выходного порта первой микроЭВМ на вход 21 блока 1 поступает сигнал $x=1$ (напряжение высокого уровня); усилитель 6, выполненный по схеме с «открытым коллектором», выключен, через резистор 7 протекает ток, светодиод оптрона 8 излучает свет, транзистор этого оптрона насыщен. Из выходного порта второй микроЭВМ на вход 32 блока 2 поступает сигнал $z=1$, так что транзистор оптрона 16 находится в состоянии насыщения. В линии связи 3 протекает ток i, достаточный для нормального свечения светодиодов оптронов 9 и 17, поэтому на их выходах сформированы сигналы $a=q=0$ (напряжения низкого уровня).

Из выходного порта первой микроЭВМ на вход 22 блока 1 в исходном состоянии поступает сигнал $h=0$, соответствующий разрешению работы формирователя импульса 11. Сигнал с на входах 25 и 26 первой микроЭВМ равен лог. 1, что соответствует отсутствию запроса на прерывание. Сигнал e в исходном состоянии имеет произвольное значение. Если $e=0$, то $f=d$; если $e=1$, то $f=\overline{d}$. Сигнал d является выходным сигналом сетевого таймера. Его частота равна 50 Гц, скважность равна двум, фаза совпадает или противоположна фазе переменного напряжения в сети 34 в силу приведенных ранее замечаний.

По сигналам от сетевых таймеров обе микроЭВМ «прерываются», т. е. временно прекращают выполнение текущих заданий и переходят к выполнению соответствующих подпрограмм, смысл которых не имеет отношения к работе устройства (например, выполняются функции часов реального времени). После выполнения подпрограмм

обе микроЭВМ возвращаются к прерванным заданиям (параметры возврата предварительно запоминаются в стековой памяти).

Первый сеанс связи может начаться при благоприятном (противофазном) или неблагоприятном (синфазном) соотношении сигналов, вырабатываемых сетевыми таймерами обеих микроЭВМ.

Работа устройства при противофазном соотношении сигналов. Сеанс связи начинается по инициативе второй микроЭВМ (микроЭВМ-приемника), которая в момент t_0 по сигналу своего сетевого таймера посылает запросный сигнал $z=0$ (рис. 5.58, 5.59). Ток в линии связи 3 прекращается, так как транзистор оптрона 16 выключается, сигналы a и q принимают единичные значения, срабатывают формирователи импульсов 11 и 12. Это приводит к прерыванию первой микроЭВМ по входу 26. (Работа микроЭВМ-передатчика и мик-

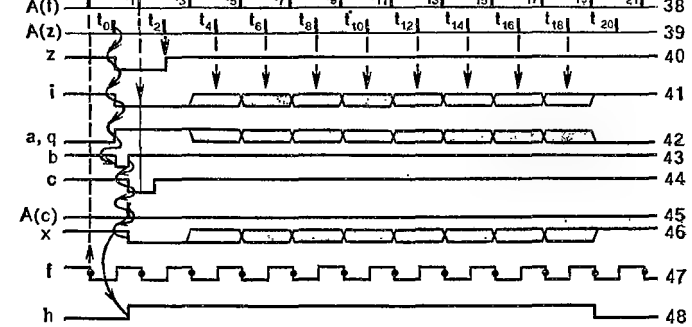


Рис. 5.58. Передача информации системой (см. рис. 5.57) при противофазной работе сетевых таймеров обеих микроЭВМ

роЭВМ-приемника по прерывающим программам осуществляется в течение коротких интервалов времени. Эти интервалы показаны на рис. 5.58 в виде «меток времени» t_0, t_1, t_2 и т. д., так как длительность каждого интервала не превышает нескольких десятков микросекунд, а масштаб рисунка не позволяет показать эти интервалы в виде «протяженных» импульсов.)

Прерывающая программа (рис. 5.60, поз. 69, 70) предписывает процессору первой микроЭВМ дополнительно разомкнуть линию связи 3 сигналом $x=0$, запретить новые прерывания по входу 26 сигналом $h=1$ (формирователь импульса 11 выключается) и принять к сведению, что при ближайшем прерывании от таймера помимо обычных действий следует выполнить действия, предписанные блоком 71 (рис. 5.60).

При получении очередного сигнала от таймера (прошедшего через элемент 10), а именно в момент t_1, при наличии отрицательного фронта сигнала f (см. точки на диаграмме 47) первая микроЭВМ выполняет действия, предписанные блоком 71. В данном случае,

благодаря выбору длительностей импульсов на выходах формирователей 11 и 12, c=0, поэтому первая микроЭВМ не меняет значение сигнала e, выдаваемого из ее выходного порта (соотношение фаз оказалось удачным).

В момент t_2 (см. рис. 5.58, 5 59) вторая микроЭВМ снимает запросный сигнал, т. е. формирует в выходном порте сигнал z=1. Ток в линии связи 3 по-прежнему отсутствует, так как транзистор оптрона 8 остается выключенным.

В момент t_3 первая микроЭВМ формирует первый информационный бит: x=0 или x=1. В зависимости от значения бита транзистор оптрона 8 выключен или включен, ток i в линии связи 3 отсутствует или имеется, сигнал q равен лог. 1 или лог. 0. В момент t_4 вторая микроЭВМ, временно переключившись на подпрограмму связи, считывает первый бит и «упаковывает» его в память. В момент t_5 первая микроЭВМ формирует второй бит, в момент t_6 вторая микроЭВМ считывает этот бит и т. д.

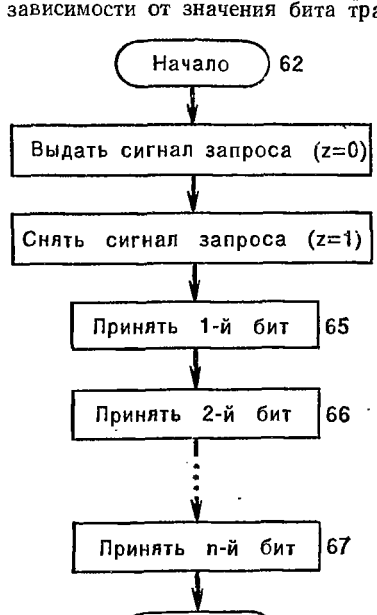


Рис. 5.59. Блок-схема алгоритма работы микроЭВМ-приемника информации (см. рис. 5.57)

Длина информационной посылки может быть любой; на рис. 5.58 показан процесс передачи одного байта (8 разрядов). После передачи последнего бита (момент t_{19}) первая микроЭВМ восстанавливает первоначальное значение сигнала h (h=0), готовясь к последующему сеансу связи. На протяжении всего сеанса связи сигнал e оставался без изменения.

Работа устройства при синфазном соотношении сигналов. В данной ситуации (рис. 5.61) сетевые таймеры обеих микроЭВМ сначала работают синфазно (см. моменты t_0 и t_1, диаграммы 50 и 51). Однако при проверке значения сигнала c (блок 71) обнаруживается, что c=1. Поэтому прерывающая программа предписывает процессору первой микроЭВМ установить сигнал e в противоположное состояние (копия сигнала e хранится в памяти первой микроЭВМ), после чего достигается нужное соотношение фаз и дальнейшая работа устройства не отличается от ранее рассмотренной.

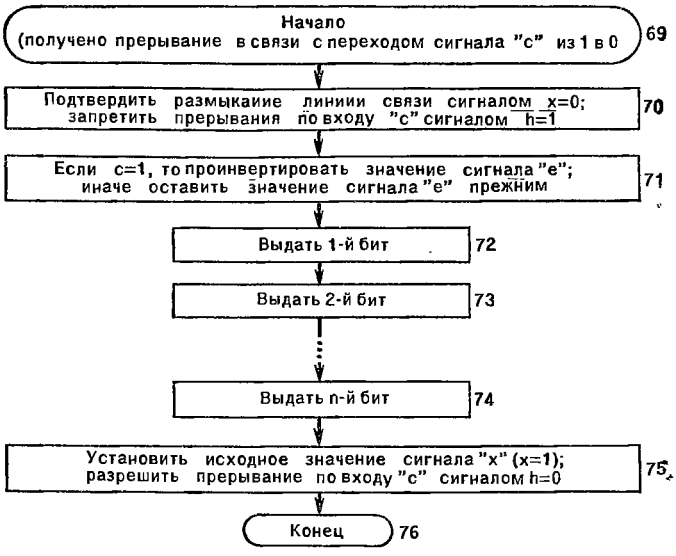


Рис. 5.60. Блок-схема алгоритма работы микроЭВМ-источника информации (см. рис. 5.57)

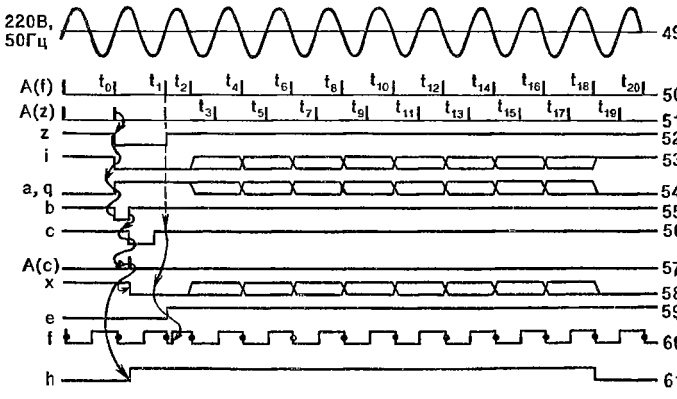


Рис. 5.61. Самосинхронизация сетевых таймеров при неблагоприятном соотношении фаз к моменту начала сеанса связи

5.24. Система опроса большого числа источников информации [154, 11]

В рассматриваемом примере система используется для управления технологическим прогоном 1024 одноплатных микроЭВМ, размещенных в 32 термокамерах, по 32 микроЭВМ в каждой. Технологический прогон является своего рода «обкаткой» микроЭВМ после их изготовления (или ремонта) и проводится в течение нескольких суток при повышенных напряжении и температуре. В процессе прогона все микроЭВМ работают в автономном режиме, выполняя тестовые программы самоконтроля, хранимые в ПЗУ. Задача заключа-

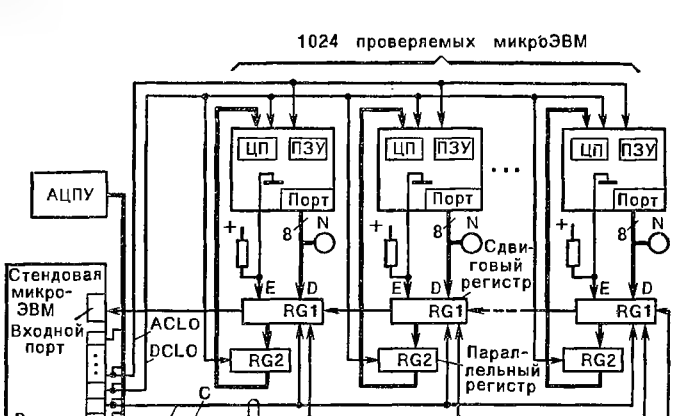


Рис. 5.62. Система сбора информации о ходе технологического прогона 1024 одноплатных микроЭВМ

ется в оперативном слежении за правильностью выполнения тестовых программ в проверяемых микроЭВМ и в обеспечении простого диалога оператора с системой.

Стендовая микроЭВМ (рис. 5.62) периодически осуществляет начальный запуск процессоров всех проверяемых микроЭВМ и следит за ходом выполнения тестовых программ в каждой проверяемой микроЭВМ.

Для выполнения начального запуска всех процессоров стендовая микроЭВМ программным путем формирует на линиях ACLO, DCLO своего выходного порта напряжения высокого уровня в соответствии с временной диаграммой, приведенной на рис. 2.7, причем напряжение питания проверяемых микроЭВМ постоянно включено. Получив сигналы запуска по двум указанным линиям, все процессоры одновременно стартуют и переходят к выполнению тестовых программ самоконтроля, хранимых в соответствующих ПЗУ. В даль-

нейшем все проверяемые микроЭВМ независимо друг от друга работают по одинаковым тестовым программам, которые построены в виде «бесконечной» циклической цепочки тестов проверки составных частей микроЭВМ.

Каждый цикл проверки по всей цепочке тестов выполняется в течение 4—5 мин в зависимости от производительности конкретной проверяемой микроЭВМ. При успешном выполнении очередного цикла проверки процессор проверяемой микроЭВМ прибавляет единицу к содержимому выходного порта (копия содержимого выходного порта хранится в ОЗУ соответствующей микроЭВМ), так что в младших четырех разрядах светодиодного индикатора N с периодом 4—5 мин появляются коды 0, 1, 2, ..., 15, 0, 1, 2 и т. д. При правильной работе микроЭВМ в одном из оставшихся четырех разрядов индикатора N информация меняется с частотой около 1 Гц, что позволяет при необходимости быстро оценивать работоспособность этой микроЭВМ, не дожидаясь завершения очередного цикла проверки (4—5 мин). Оставшиеся разряды отображают служебные признаки, например номер модификации микроЭВМ.

Стендовая микроЭВМ периодически, например через каждые 6—7 мин, опрашивает состояние выходных портов всех проверяемых микроЭВМ. Опрос осуществляется параллельным занесением информации из выходных портов проверяемых микроЭВМ в регистры RG1 с последующим поразрядным сдвигом этой информации во входной порт стендовой микроЭВМ, которая принимает полученные в результате сдвига биты и упаковывает их в свое ОЗУ. Полученная информация представляет собой «фотографию» состояния всех светодиодных индикаторов N. Сопоставляя эти «фотографии», полученные с интервалом 6—7 мин, стендовая микроЭВМ имеет возможность следить за динамикой изменения кодов в выходных портах всех проверяемых микроЭВМ.

При правильной работе проверяемой микроЭВМ в четырех младших разрядах ее выходного порта к моменту очередного опроса код должен измениться, так как к нему за время между опросами прибавилась одна или две единицы.

Сигнал E служит указателем наличия проверяемой одноплатной микроЭВМ в данном посадочном месте: если плата установлена, то E=0, если нет, то E=1. Поэтому, анализируя признаки E, стендовая микроЭВМ «узнаёт», какие коды D достоверны, а какие не имеют смысла.

Для опроса состояния проверяемых микроЭВМ стендовой микроЭВМ выполняются следующие действия (используется «программируемый интерфейс», см. § 2.4).

1. В линию Прием/сдвиг, определяющую режим работы сдвиговых регистров RG1, выдается напряжение высокого уровня (H). Регистры RG1 подготовлены к параллельному приему информации.

2. В линии C создается импульс (программным путем). При положительном перепаде напряжения в этой линии в сдвиговые регистры RG1 принимается новая информация от проверяемых плат.

3. В линию Прием/сдвиг выдается напряжение низкого уровня (L). Регистры RG1 подготовлены к сдвигу информации на один разряд влево по положительным фронтам сигнала в линии C.

4. Через входной порт стендовой микроЭВМ считывается сигнал

из левого разряда ближайшего сдвигового регистра RG1. Соответствующий бит упаковывается в ОЗУ стендовой микроЭВМ.

5. Если введена не вся информация из цепи сдвиговых регистров RG1, то в линии C создается импульс и производится переход к п. 4. Если введена вся информация, то процесс опроса завершается.

По мере освобождения разрядов цепи из сдвиговых регистров RG1 эти разряды заполняются тестовой информацией, которая генерируется стендовой микроЭВМ. Тестовая информация с линии S проходит по всей последовательной цепи из сдвиговых регистров RG1 и возвращается в стендовую микроЭВМ, где сравнивается с исходной. При обнаружении ошибкн, например обрыва цепи из сдвиговых регистров, оператору выдается сообщение о случившемся и он принимает соответствующие меры по устранению неисправности.

При работе системы может оказаться, что в момент записи новой информации из проверяемых плат в сдвиговые регистры RG1 эта информация меняется. Тогда в одном или нескольких регистрах может быть зафиксирована промежуточная информация — не старая, но еще и не новая. Стендовая микроЭВМ, зная динамику изменения кодов D при правильной работе проверяемой микроЭВМ, может отличить промежуточную информацию от ожидаемой и в необходимых случаях повторить опрос подозреваемых в неправильной работе плат для выяснения истины.

Перед выполнением начального запуска проверяемых микроЭВМ стендовая микроЭВМ загружает регистры RG1 некоторой информацией, используя линии S, C, Прием/сдвиг. Эта информация в процессе начального запуска проверяемых микроЭВМ (по положительному фронту сигнала на линии DCLO) переписывается в регистры RG2 и остается стабильной вплоть до момента следующего запуска, который осуществляется, например, после приостановки системы для снятия с прогона отказавших микроЭВМ, установки новых и т. п. Информация из регистров RG2 поступает в соответствующие микроЭВМ и определяет режим их работы: с включенным или выключенным корректором ошибок ОЗУ, с включенным или отключенным таймером и т. п. Режим работы периодически, например ежечасно, меняется, чтобы контроль был более полным.

В каждой термокамере предусмотрено дополнительное посадочное место, которое не используется по прямому назначению и служит для задания номера термокамеры, который устанавливается на линиях D с помощью микропереключателей или паяных перемычек. Это позволяет поручить программе (выполняемой стендовой микроЭВМ) слежение за конфигурацией системы при исключении из нее одной или нескольких термокамер для ремонта, введении новых термокамер и т. п.

Выходной порт стендовой микроЭВМ используется также и по прямому назначению — для связи с печатающим устройством (АЦПУ), при этом задействованы те же разряды порта, что и занятые для обслуживания системы сбора информации. Это приводит к тому, что при выдаче данных из стендовой микроЭВМ на печать система сбора информации переходит в неопределенное состояние. Поэтому после выдачи данных на печать стендовая микроЭВМ производит начальный запуск всех проверяемых одноплатных устройств, как было описано ранее.

С другой стороны, чтобы печатающее устройство не давало ложных срабатываний при нормальной работе системы сбора ин-

формации, возникающие кодовые комбинации на линиях ACLO, DCLO, Прием/сдвиг, C, и S не должны им распознаваться. Например, некоторые печатающие устройства выполняют печать строки только при распознании кода команды Возврат каретки или Перевод строки, остальные коды не приводят к ложным срабатываниям. Таким образом, при проектировании системы сбора информации достаточно исключить возможность появления двух указанных кодов в режиме обслуживания проверяемых одноплатных устройств.

Отметим, что для правильной работы цепи из сдвиговых регистров RG1 необходимо, чтобы сигнал C, проходя по цепи ретрансляторов, распространялся навстречу сдвигаемой информации, иначе информация при сдвиге будет искажаться. В системе предусмотрены отладочные режимы работы для облегчения поиска неисправностей с точностью до микросхемы регистра RG1 и RG2.

Программное обеспечение системы позволяет во многом автоматизировать операцию технологического прогона одноплатных устройств. Это, в свою очередь, позволяет повысить объективность контроля и исключить ошибки оператора (непреднамеренные или преднамеренные), в результате которых ненадежно работающие устройства квалифицируются как годные. Более того, система следит за качеством работы операторов (например, за своевременностью снятия с прогона отказавших устройств, за отсутствием попыток досрочного снятия с прогона годных устройств и т. п.) и проставляет им оценки. Каждый оператор имеет свой код, по которому система его распознаёт.

Программное обеспечение системы предоставляет оператору ряд «меню» для получения интересующей его информации. В частности, оператор может узнать о наличии свободных посадочных мест в термокамерах, о состоянии проверяемых устройств, получить прогноз выхода с прогона годных устройств на ближайшую смену или на ближайшие часы, выявить платы (по заводским номерам), которые многократно попадали на прогон и снимались с прогона из-за ненадежной работы и т. п. Стендовая микроЭВМ периодически (например, ежечасно) записывает всю текущую информацию системы на жесткий магнитный диск, чтобы не произошло невосполнимых потерь данных при отказе микроЭВМ, аварийном отключении напряжения первичной питающей сети и т. п.

5.25. Повышение производительности микроЭВМ при рассасывании очередей из запросов на прерывания [118, 119, 203]

Если несколько внешних (по отношению к микроЭВМ) абонентов одновременно требуют обслуживания со стороны центрального процессора, то образуется очередь. Так как процессор не может одновременно обслужить все имеющиеся запросы, он принимает один из запросов, обслуживает его, затем принимает один из оставшихся и т. д., вплоть до полной ликвидации очереди. (Очередь может пополняться, пока процессор пытается ее ликвидировать.) Порядок обработки запросов может быть любым, в зависимости от особен-

ностей построения конкретной микроЭВМ. Например, запросы могут обрабатываться в порядке поступления (FIFO), в порядке, обратном очередности поступления (LIFO), в порядке возрастания или убывания номеров линий связи, по которым поступили сигналы запросов, и т. п.

Далее предполагается, что запросы обрабатываются строго поочередно, т. е. одна прерывающая программа не может прервать другую. Это предположение принято лишь для того, чтобы более наглядно проявились преимущества предлагаемых решений по сравне-

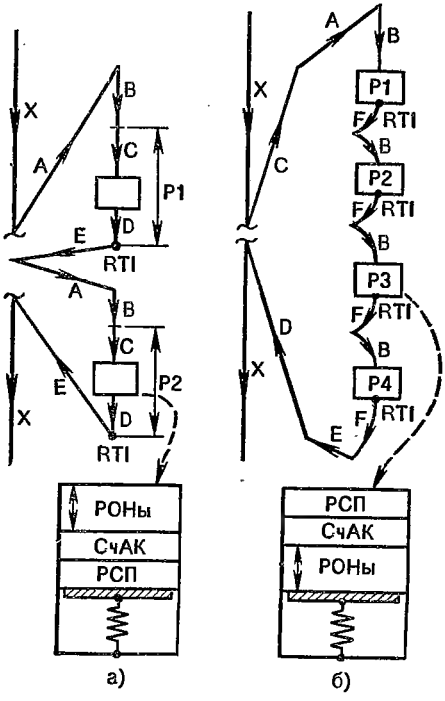


Рис. 5.63. Варианты построения систем прерывания.

При обработке очереди из запросов на прерывания микроЭВМ типа «Электроника-60» возвращается к фоновой задаче после обработки очередного запроса (*а*); предлагается не возвращаться к фоновой задаче до тех пор, пока не будет ликвидирована очередь (*б*). Штриховые линии указывают на содержимое стека в соответствующие моменты времени

нию с традиционными. Однако следует отметить, что преимущества теряются при «вложении» прерывающих программ, когда одна прерывающая программа прерывает выполнение другой, менее приоритетной.

В двух рассмотренных далее примерах повышение производительности микроЭВМ достигается благодаря быстрому переключению процессора с одной прерывающей программы на другую — при таком переключении процессор не пытается вернуться к выполнению фоновой программы до тех пор, пока не будет полностью ликвидирована очередь из запросов на прерывания.

Пример 1 [118]. На рис. 5.63, *а* приведена блок-схема широко распространенного алгоритма обработки прерываний, который ис-

пользуется, например, в микроЭВМ типа «Электроника-60». Рассмотрим сначала этот алгоритм, чтобы понять его недостатки и затем перейти к предлагаемому алгоритму, показанному на рис. 5.63, *б*.

В исходном состоянии центральный процессор микроЭВМ выполняет фоновую программу X (рис. 5.63, *а*). При получении первого (разрешенного) запроса на прерывание процессор прекращает выполнение фоновой программы, посылает в магистраль сигнал предоставления прерывания, получает от абонента вектор и переходит к процедуре А — запоминанию в стеке содержимого регистра состояния процессора и счетчика адреса команды (соответственно РСП и СчАК). В регистре состояния процессора содержатся признаки, зависящие от результата выполнения предыдущих команд («переполнение», «отрицательный результат» и др.), признаки, определяющие приоритет процессора по отношению к возможным прерываниям (эти признаки определяют «степень защиты» фоновой программы от «мешающих» прерываний), и др. Счетчик адреса команды служит указателем адреса ячейки памяти с размещенной в ней командой, которой передается управление при возврате процессора к фоновой программе X.

После выполнения процедуры А процессор автоматически переходит к выполнению процедуры В. На данном этапе процессор загружает счетчик адреса команды и регистр состояния информацией из двух соседних 16-разрядных ячеек ОЗУ, выбираемых по вектору прерывания, полученному от абонента, требующего обслуживания и стоящего первым на очереди. Например, если вектор указывает на ячейку ОЗУ с адресом 320, то код из этой ячейки помещается в счетчик адреса команды (старое значение сохранено в стеке), а код из соседней ячейки (с адресом 322) — в регистр состояния процессора. Для строго поочередного выполнения прерывающих программ новое содержимое регистра состояния предусматривает запрет реакции на другие требования прерывания, которые стоят на очереди. После выполнения процедур А и В процессор приступает к выполнению первой команды прерывающей программы P1.

В примере, приведенном на рис. 5.63, *а*, предполагается, что прерывающая программа предусматривает сохранение в стеке (процедура С) содержимого регистров общего назначения процессора (РОН). Это делается для того, чтобы освободить РОН для использования в ходе выполнения прерывающей программы (работа с РОН позволяет ускорить ее выполнение).

После выполнения «полезной» части прерывающей программы P1 (эта часть показана на рис. 5.63, *а* прямоугольником) процессор восстанавливает из стека содержимое РОН, соответствующее фоновой программе (процедура D). Восстановление содержимого РОН, так же как и его сохранение (процедура С), выполняется программно. Иными словами, сохранность исходного содержимого РОН при выходе из прерывающей программы обеспечивается самой прерывающей программой (процедуры С и D).

После восстановления содержимого РОН процессор извлекает из памяти последнюю команду прерывающей программы — команду возврата из прерывания (RTI). В ходе выполнения этой команды (процедура Е) из стека извлекается содержимое счетчика адреса команды и регистра состояния процессора. Восстановленное «старое» содержимое регистра состояния процессора предусматривает разрешение обработки очередного запроса на прерывание. Поэтому процессор вновь посылает источнику сигнал предоставления прерывания,

получает от него вектор, выполняет процедуры A, B и переходит к выполнению прерывающей программы P2. Далее все описанные ранее действия повторяются до тех пор, пока не будет обслужен последний запрос на прерывание, после чего процессор возвращается к выполнению фоновой программы.

Недостаток рассмотренного алгоритма состоит в том, что процессор при переходе от одной прерывающей программы к другой выполняет много лишних действий. Он восстанавливает содержимое своих регистров, а затем, не пользуясь результатом восстановления, вновь записывает информацию из этих же регистров на старое место — в стек. Таким образом, последовательность процедур D — E — A — B — C теряет смысл и ее следовало бы исключить (что фактически и предлагается сделать).

Предлагаемый алгоритм (рис. 5.63, *б*) содержит процедуры, аналогичные или близкие рассмотренным (рис. 5.63, *а*); они имеют соответствующие обозначения: A, B, C, D, E. Однако последовательность выполнения этих процедур отличается от рассмотренной. Кроме того, введена новая процедура F.

В исходном состоянии процессор работает по фоновой программе X (рис. 5.63, *б*). При обнаружении запроса на прерывание микропрограммно выполняются процедуры C, A и B — запоминание содержимого РОН в стеке (C), запоминание содержимого СчАК, РСП (A) и загрузка СчАК и РСП новыми кодами, выбираемыми по вектору (B). При этом новое содержимое РСП защищает прерывающую программу от прерываний, стоящих на очереди. Прерывающая программа P1 в данном случае содержит только «полезную» часть, непосредственно ориентированную на обслуживание соответствующего абонента: РОН свободны, так как их старое содержимое уже сохранено в стеке.

Программа P1, так же как и показанная на рис. 5.63, *а*, завершается командой выхода из прерывания RTI, однако исполняется эта команда по-иному.

Процессор, выполняя процедуру F, микропрограммно восстанавливает старое значение кода в регистре состояния, считывая этот код из верхней ячейки стека. При считывании используется метод адресации, не меняющий код в регистре — указателе стека, т. е. информация из стека не «выталкивается». Восстановленный код в регистре состояния, как уже отмечалось, разрешает обрабатывать очередной запрос на прерывание, если очередь еще не ликвидирована.

Если аппаратура процессора обнаружила запрос на прерывание, то, как обычно, абоненту посылается сигнал предоставления прерывания, с шины данных считывается вектор и из двух соседних ячеек ОЗУ (выбираемых по вектору) извлекается новое содержимое счетчика адреса команды и регистра состояния (процедура B). Необходимости сохранения информации нет, так как эта информация уже хранится в стеке. Если после выполнения процедуры F аппаратура процессора не обнаружила запросов на прерывание, разрешенных содержимым регистра состояния, то производится возврат к фоновой программе X. При этом выполняются процедуры E и D. Процедура E предусматривает только восстановление содержимого СчАК и коррекцию кода в регистре — указателе стека, так как содержимое регистра состояния процессора уже восстановлено при выполнении процедуры F. Процедура D (восстановление содержимого РОН) выполняется в полном объеме. Процедуры F — B или

F — E — D ныполняются микропрограммно при исполнении команды RTI.

Сравнивая известный и предлагаемый алгоритмы, рис. 5.63, *а, б*, отметим, что параметры возврата к фоновой задаче к моменту завершения обслуживания абонента размещены в стеке по-разному, а именно во взаимно-обратном порядке. С учетом исключения «лишних» операций время переключения с одной прерывающей программы на другую сокращено примерно в 10 раз.

Пример 2 [119, 203]. В микроЭВМ (рис. 5.64) введена дополнительная аппаратура — дешифратор, мультиплексор и элемент И. Эта аппаратура не дает возможности процессору, вошедшему в ре-

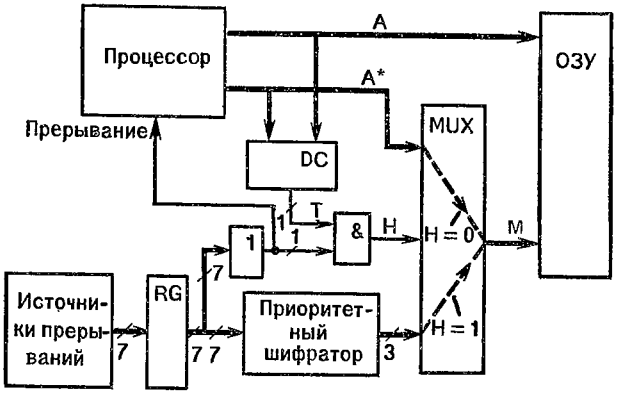


Рис. 5.64. Схема обработки сигналов прерывания.

При дешифрации кода 2000 и наличии очереди мультиплексор MUX передает на выходы M код с выходов R приоритетного шифратора

жим прерывания, вернуться к фоновой задаче до тех пор, пока не будет полностью ликвидирована очередь из запросов на прерывания. Процессор при этом «не подозревает» о существовании «ловушки» (построенной на дополнительной аппаратуре), в которую он попадает при первоначальном переходе в обычный режим прерывания в ответ на один из сигналов запроса. С «точки зрения» процессора он обрабатывает однократный запрос, т. е. работает по одной прерывающей программе. В действительности он последовательно выполняет целую цепочку из прерывающих программ, стоящих в очереди.

В отсутствие прерываний процессор работает по фоновой программе. На выходе дешифратора DC сформирован сигнал лог. 0, элемент И закрыт. Сигнал H=0 поддерживает мультиплексор в состоянии, при котором его выходы M соединены с входами A*, которые представляют собой группу младших разрядов адресной шины процессора. Группа A старших разрядов адресной шины соединена с соответствующими адресными входами ОЗУ. Таким образом, в отсутствие прерываний старшие и младшие разряды адресной шины

процессора соединены с соответствующими адресными входами ОЗУ. Шина данных на рис. 5.64 не показана.

При поступлении одного или нескольких сигналов запросов на прерывание от источников эти сигналы запоминаются в регистре. На выходе элемента ИЛИ формируется сигнал лог. 1, который вызывает прерывание процессора и подается на первый вход элемента И. Приоритетный шифратор формирует на выходах двоичный код, соответствующий номеру наиболее приоритетного источника прерывания из числа требующих обслуживания.

Процессор запоминает параметры возврата к фоновой программе и переходит к прерывающей программе по фиксированному вектору, не зависящему от кодовой комбинации в регистре RG. Эта прерывающая программа содержит две команды:

1) команду JUMP безусловной передачи управления по адресу 5000, который хранится в ячейке ОЗУ с адресом 2000 (коды выбраны в качестве примера);

2) команду RTI (выход из прерывания), которая хранится в ячейке ОЗУ с адресом 5000.

При выполнении команды (1) на адресной шине формируется код 2000. Дешифратор DC реагирует на этот код выдачей сигнала T=1. В результате срабатывает элемент И (H=1) и с помощью мультиплексора MUX код с младших разрядов адресной шины процессора подменяется кодом с выходов приоритетного шифратора. Например, вместо кода 2000 на адресные входы ОЗУ могут быть поданы коды 2002, 2004, 2006, 2010, 2012, 2014, 2016 в зависимости от кода, сформированного приоритетным шифратором.

Поэтому процессор фактически выполняет команду безусловного перехода по адресу, который хранится в одной из семи указанных ячеек ОЗУ (2002—2016). В этих ячейках записаны начальные адреса программ обслуживания соответствующих запросов на прерывания, так что процессор переходит на начало нужной программы и приступает к ее выполнению. При этом соответствующий сигнал запроса в регистре RG снимается (устанавливается в 0), например, программным путем, если регистр выполнен программно-доступным (см. гл. 1).

Завершающей командой каждой программы обслуживания запроса на прерывание является описанная ранее команда JUMP безусловной передачи управления по адресу 5000, который хранится в ячейке ОЗУ с адресом 2000. При выполнении этой команды возможны две ситуации: (а) очередь еще не ликвидирована и (б) очередь отсутствует.

В обеих ситуациях адрес 2000 распознаётся дешифратором DC и с помощью элемента И проверяется наличие необработанных запросов на прерывания. Если таковые имеются (ситуация а), то H=1, мультиплексор соответствующим образом корректирует младшие разряды адресного кода 2000, как было показано ранее, и процессор приступает к обслуживанию очередного источника прерывания. Если очереди нет (ситуация б), то H=0 и код 2000 поступает на адресные входы ОЗУ без искажения (выходы мультиплексора соединены с входами A*). Процессор извлекает из ячейки с адресом 2000 код 5000, который, в свою очередь, является адресом команды, которой передается управление. Как уже отмечалось, в ячейке с адресом 5000 хранится код команды RTI, так что процессор, исчерпан очередь, возвращается к выполнению фоновой программы.

На рис. 5.65 приведена диаграмма, поясняющая описанный прин-

цип обработки очереди из запросов на прерывания. На рис. 5.65 приняты следующие обозначения:

X — фоновая программа;

A° — запоминание в стеке параметров возврата к фоновой программе;

B — загрузка счетчика адреса команды и регистра состояния процессора информацией из двух ячеек ОЗУ, выбираемых по вектору прерывания;

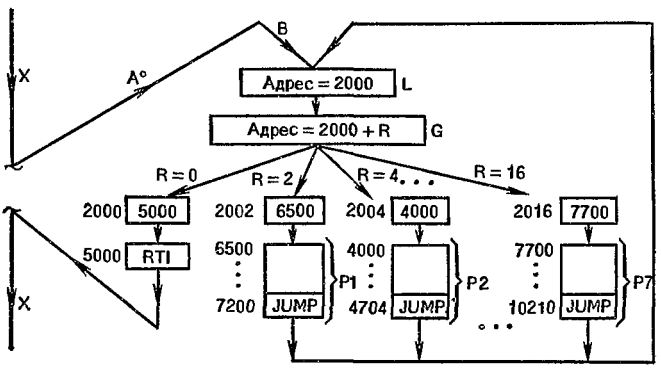


Рис. 5.65. Работа процессора при обработке сигналов прерывания.

Процессор, отложив выполнение фоновой программы X, не вернется к ней до тех пор, пока не будут исчерпаны работы по прерывающим программам P_I, стоящим в очереди

L — выдача в адресную шину процессора кода 2000;

G — модификация адреса мультиплексором (R=0, 2, 4, 6, 10, 12, 14, 16);

P_i — прерывающие программы.

Отметим, что рассмотренное схемное решение (рис. 5.64) до некоторой степени схоже с приведенным в § 5.4, где также используется подмена одного кода другим.

5.26. Параллельный опрос ряда источников векторных прерываний (применительно к каналу микроЭВМ типа «Электроника-60»)

На рис. 5.66 показана схема подключения ряда контроллеров к каналу микроЭВМ типа «Электроника-60», позволяющая уменьшить время реакции на прерывание за счет параллельного (а не последовательного) схемного опроса.

В исходном состоянии центральный процессор (не показан) выполняет фоновую программу, контроллеры $K1$—K_N не требуют обслуживания, регистр RG находится в режиме трансляции сигналов с входов на выходы.

В некоторый момент один или несколько контроллеров формируют сигналы $Q_i = L$ (запросы на прерывания, представленные напряжениями низкого уровня). Арбитр выделяет наиболее приоритетный сигнал, который проходит в линию ТПР (требование прерывания) и поступает в процессор. В ответ на сигнал ТПР процессор формирует сигнал Ввод, который запрещает формирование «новых» сигналов Q_i («старые» по-прежнему воздействуют на арбитр).

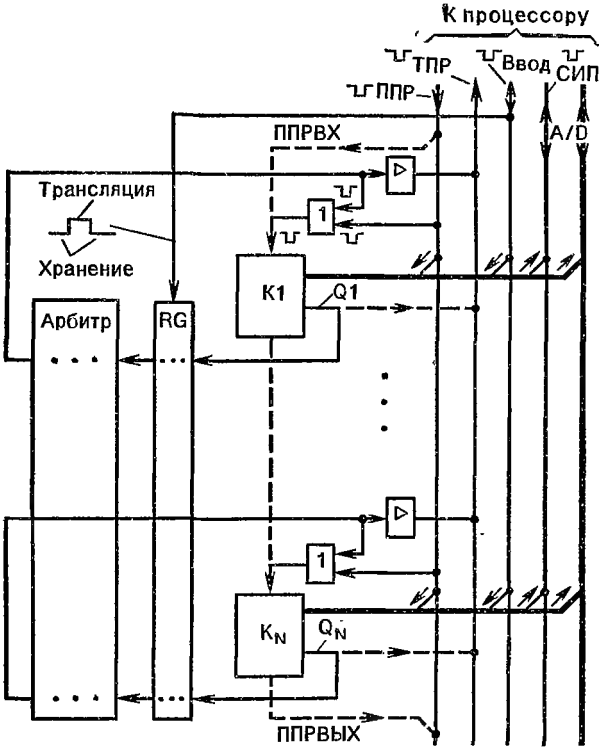


Рис. 5.66. Схема опроса контроллеров.

В отличие от традиционной схемы включения контроллеров (дейзи-цепь), используется схема, содержащая дополнительную аппаратуру: регистр, арбитр, логические элементы; это позволило перейти от последовательной цепи распространения сигнала ППР к параллельной

После окончания возможных переходных процессов на выходах Q_i, через 150 нс (или более) после формирования сигнала Ввод, процессор формирует сигнал ППР (предоставление прерывания), регистр RG переходит в режим хранения имеющейся в нем информации. Сигнал ППР воздействует одновременно на все элементы ИЛИ, но проходит только через один из них на вход выбранного арбитром контроллера.

Далее, как обычно, из контроллера выдается адрес вектора прерывания и ответный сигнал СИП (синхросигнал пассивного устройства). Процессор принимает адрес вектора и снимает сигналы Ввод и ППР. Регистр RG переходит в режим трансляции входных сигналов и если нет других запросов (обслуженный запрос к этому времени снят), сигнал ТПР снимается. Контроллер освобождает шину адресов — данных A/D и снимает сигнал СИП.

Если имеются необслуженные запросы, то в линии ТПР поддерживается напряжение низкого уровня, т. е. создается очередь из запросов.

Штриховые линии на рис. 5.66 соответствуют обычной схеме с последовательным распространением сигнала ППР через ряд контроллеров.

5.27. Об одной особенности подключения программно-доступных элементов к каналу микроЭВМ типа «Электроника-60»

На рис. 5.67 приведен пример некорректного построения схемы, обеспечивающей программный доступ к триггеру Тг2. Приведенное далее описание составлено так, чтобы убедить вас в том, что схема работает правильно. Поэтому постарайтесь обнаружить ошибку самостоятельно, прежде чем обратиться к разъяснениям.

При обращении к программно-доступному триггеру Тг2 со стороны центрального процессора или канала прямого доступа на совмещенной шине адресов — данных A/D устанавливается некоторый адрес, который распознаётся дешифратором DC. Истинность адреса подтверждается отрицательным фронтом сигнала СИА (синхросигнал активного устройства), который вырабатывается через 150 нс (или более) после выдачи адреса в шину A/D (как в канале Общая шина, см. рис. 1.6). Результат дешифрации запоминается в триггере Тг1: если адрес распознан как «наш», то триггер Тг1 устанавливается в состояние лог. 1; если адрес не распознан (обращение не относится к триггеру Тг2), то триггер Тг1 устанавливается в состояние лог. 0.

После передачи адреса активное устройство освобождает шину A/D и в зависимости от типа операции (запись или чтение) выдает в шину A/D данные и с задержкой 100 нс или более сигнал Вывод (при записи) или, не занимая шину A/D, сигнал Ввод (при чтении). Если адрес был предварительно распознан (триггер Тг1 был установлен в состояние лог. 1), то сигнал Вывод воздействует на вход синхронизации С триггера Тг2, в результате информация из нулевого разряда шины A/D записывается в этот триггер; при чтении сигнал Ввод опрашивает состояние триггера Тг2 и через элемент И — НЕ в нулевой разряд шины A/D поступает соответствующий бит.

Если адрес был предварительно распознан, то при поступлении сигналов Ввод или Вывод срабатывает элемент ИЛИ — НЕ и из нашего контроллера выдается сигнал СИП (синхросигнал пассивного устройства). Этот сигнал подтверждает, что обращение прошло нормально и требуемая операция (Вывод или Ввод) выполнена. Вводить какие-либо задержки в цепь формирования сигнала СИП или в другие цепи не нужно — временные диаграммы работы этой схемы в полной мере отвечают соответствующим требованиям. Сигнал

Сброс вырабатывается в двух ситуациях: при включении напряжения питания и при выполнении центральным процессором команды RESET. Во всех остальных ситуациях сигнал Сброс отсутствует и не мешает нормальной работе контроллера. При необходимости можно сигналом Сброс устанавливать в 0 триггер Тг2, объединив R-входы обоих триггеров.

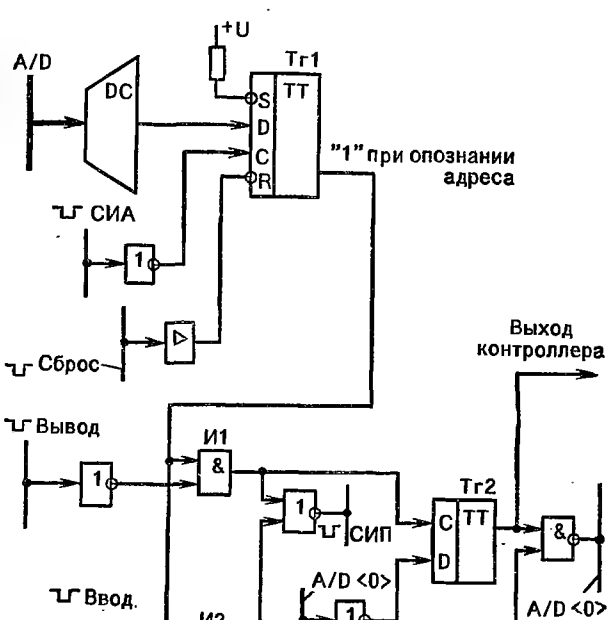


Рис. 5.67. Пример некорректного подключения программно-доступного триггера Тг2 к каналу микроЭВМ типа «Электроника-60» (типичная ошибка начинающих схемотехников!)

Разъяснения. Ошибка проявляется в тех случаях, когда после обращения к нашему контроллеру следует прерывание центрального процессора со стороны «чужого» контроллера. Прерывание сопровождается сигналом Ввод без сигнала СИА, а наш триггер Тг1 все еще помнит факт опознания из предыстории. Поэтому из нашего контроллера в ответ на сигнал Ввод (этот сигнал используется для предотвращения гонок в последовательной цепи распространения сигнала предоставления прерывания, см. § 5.26) будет несанкционированно выдан ответный сигнал СИП и бит данных из триггера

Тг2. Простейший вариант доработки схемы (рис. 5.67) — соединить вход С триггера Тг1 с дополнительно введенным третьим входом элемента И2.

5.28. Выявление конфигурации управляемого объекта с использованием диодной схемы ИЛИ [155]

При использовании микроЭВМ для управления объектом в ряде случаев необходимо уточнить тип объекта, чтобы микроЭВМ «знала» о его особенностях и пользовалась соответствующей программой, хранимой в ее ПЗУ. Например, одна и та же микроЭВМ может использоваться для управления дверьми автобуса, трамвая, троллейбуса. Программы управления этими объектами имеют общие части, но учитывают особенности конкретной системы: число задействован-

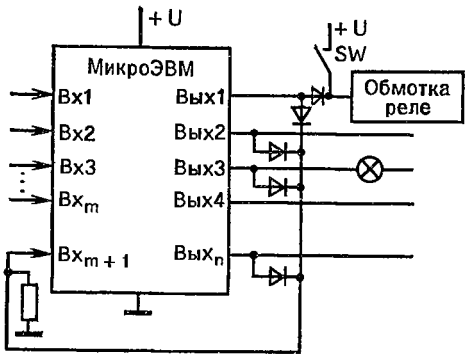


Рис. 5.68. Подключение микроЭВМ к управляемому объекту с возможностью опроса диодной схемы, задающей тип объекта

ных входов, выходов микроЭВМ, их функциональное назначение, условия экстренного открывания дверей, оптимальные режимы управления и т. п.

Введение k дополнительных внешних входов в микроЭВМ позволяет путем соответствующей «распайки» задать 2^k кодовых комбинаций, каждая из которых соответствует определенному типу объекта. После включения напряжения питания микроЭВМ опрашивает эти входы и по результатам опроса приступает к выполнению соответствующей рабочей программы. Такой способ, однако, неэкономичен. Предлагается уменьшить число дополнительных внешних входов до одного и проводить последовательный опрос выходов, к которым подключена диодная схема ИЛИ (рис. 5.68).

Входы Вх1 — Вх$_m$ и выходы Вых1 — Вых$_n$ микроЭВМ предназначены для ее связи с управляемым объектом, вход Вх$_{m+1}$ служит для сбора информации с выходов в период начального опроса при выявлении типа объекта. К выходам подключены исполнительные

органы — обмотки реле, индикаторы и т. п., к входам — клавиши пульта управления, концевые выключатели, датчики давления и т. п. Ключ SW подключен к обмотке реле «в обход» микроЭВМ через развязывающий диод (этот диод не входит в состав схемы ИЛИ), что позволяет, например, при отказе микроЭВМ управлять объектом вручную.

После включения напряжения питания микроЭВМ формирует на выходах однократную серию кратковременных импульсов положительной полярности (сигнал типа «бегущая единица»). Длительность этих импульсов настолько мала, что они не успевают воздействовать на управляемый объект. При наличии диода на соответствующем выходе (Вых1 — Вых3, $Вых_n$) положительный импульс передается на вход опроса $Вх_{m+1}$, при отсутствии диода (Вых4) на этом входе поддерживается напряжение низкого уровня. МикроЭВМ ведет учет считанных бит и в результате формирует n-разрядный код, в котором содержится информация о наличии диодов и их местоположении. Таким образом, с помощью «распайки» диодов можно задать 2^n различных кодовых комбинаций, соответствующих различным типам объектов.

5.29. Поиск источника прерывания с использованием группового опроса контроллеров [162]

Метод программного последовательного опроса (§ 2.5, рис. 2.32) позволяет выявлять контроллер, пославший в микроЭВМ сигнал прерывания. Чтобы ускорить поиск, предлагается производить групповой опрос состояния контроллеров, рис. 5.69.

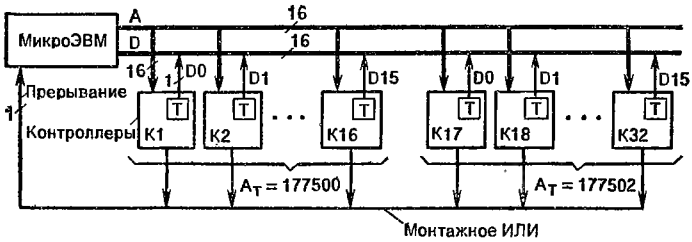


Рис. 5.69. Система с групповым опросом триггеров Т, информация в которых характеризует состояние контроллеров.

A_T — адрес триггера при его опросе со стороны микроЭВМ

В каждом контроллере имеется программно-доступный триггер Т запроса на прерывание. Этот триггер устанавливается в 1 внутренними схемами контроллера при необходимости его обслуживания со стороны центрального процессора микроЭВМ. Сигналы с триггеров Т суммируются схемой Монтажное ИЛИ и результирующий сигнал поступает в центральный процессор микроЭВМ. Особенность системы в том, что все триггеры Т разных контроллеров распределены по группам. Число триггеров в каждой группе не превышает

число линий шины данных D магистрали микроЭВМ, при этом все триггеры группы выдают информацию в разные линии шины данных. Каждая группа триггеров имеет единый адрес, так что при считывании информации по этому адресу в разные разряды шины данных микроЭВМ одновременно выдается содержимое всех триггеров Т данной группы.

В примере, приведенном на рис. 5.69, контроллеры К1 — К16 образуют первую группу, в которой триггеры Т доступны по чтению со стороны микроЭВМ по адресу 177500_8; контроллеры К17 — К32 образуют вторую группу с адресом чтения триггеров Т, равным 177502, и т. д. Триггер Т контроллера К1 при считывании выдает один бит информации в линию D0 шины данных, остальные триггеры первой группы в это же время выдают информацию в линии D1 — D15. Триггеры Т контроллеров К17 — К32 при обращении центрального процессора микроЭВМ ко второй группе также выдают информацию в линии D0 — D15 и т. д. Таким образом, при выполнении одной команды чтения центральный процессор микроЭВМ производит опрос состояния группы из 16 контроллеров.

Остальные программно-доступные элементы контроллеров (не показаны на рисунке) имеют не «групповые», а обычные «индивидуальные» адреса, так что центральный процессор микроЭВМ имеет возможность индивидуальной работы с выбранным контроллером.

В исходном состоянии микроЭВМ выполняет задание, не связанное с обслуживанием контроллеров, все триггеры Т установлены в 0. При возникновении условий прерывания один или несколько триггеров Т устанавливаются в 1, в центральный процессор поступает обобщенный сигнал Прерывание. Центральный процессор микроЭВМ временно прекращает выполнение текущего задания, запоминает параметры возврата и переходит к выполнению прерывающей программы. В начале этой программы предусмотрено считывание данных из ячеек памяти с адресами 177500, 177502 и т. д. В качестве этих ячеек используются соответствующие наборы триггеров Т. Указанные адреса поступают из центрального процессора микроЭВМ в адресную шину А и дешифрируются соответствующими группами контроллеров, так что информация о состоянии контроллеров в быстром темпе пересылается в процессор для последующего программного анализа и выявления контроллера-претендента на первоочередное обслуживание.

Ответные управляющие сигналы, выдаваемые разными контроллерами при групповом опросе триггеров Т, суммируются на общей линии ответа шины управления (на рисунке не показана). Поэтому задержка выдачи ответного сигнала самым «быстрым» контроллером должна быть выбрана достаточно большой, чтобы самый «медленный» контроллер успел выдать содержимое триггера Т (0 или 1) в соответствующую линию шины данных.

5.30. Клавиатурный блок дистанционного управления с низким потреблением энергии в паузах между сеансами связи [165]

Блок (рис. 5.70) содержит матричную клавиатуру К (кружка́ми обозначены нормально-разомкнутые ключи), процессор типа 80С48 (ф. NEC), передающий светодиод, излучающий инфракрасный свет, одновибратор F, триггеры D1, D2, D3 — D4 и другие элементы. Блок

предназначен для дистанционного управления приборами — телевизором, радиоприемником, компьютером и т. п.; его питание осуществляется от батареи или аккумулятора. Управляемый прибор (не показан на рисунке) содержит фотоприемник инфракрасного излучения и дешифратор принятого кода.

Особенность данной схемы в том, что в пассивном состоянии она потребляет значительно меньший ток, чем при работе с клавиатурой. Триггеры, одновибратор, инверторы и усилитель выполнены на

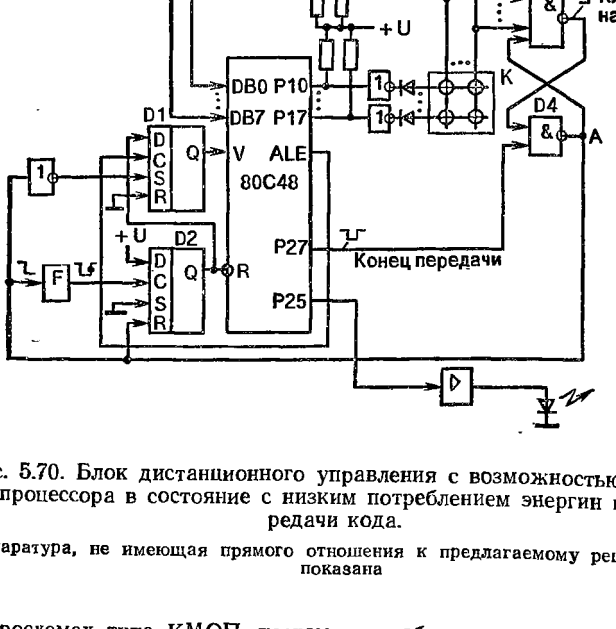


Рис. 5.70. Блок дистанционного управления с возможностью перевода процессора в состояние с низким потреблением энергии после передачи кода.

Аппаратура, не имеющая прямого отношения к предлагаемому решению, не показана

микросхемах типа КМОП, поэтому потребление тока этими элементами в статическом режиме не превышает долей микроампера. Процессор типа КМОП в зависимости от сигнала на входе V может находиться в активном либо пассивном состоянии. При V=1 (активное состояние) потребление тока процессором составляет 6 мА, при V=0 (пассивное состояние) — не более 1,5 мкА.

Если в течение 8 ч человек-оператор совершает 10 тыс. нажатий на клавиши, то суммарная доля времени пребывания блока в активном состоянии составит 3,5 %. Суммарный электрический заряд, отбираемый в течение 8 ч от батареи, — 1,7 мА·ч, что примерно в 30 раз меньше, чем без периодического перевода процессора в режим низкого потребления энергии. Рассмотрим работу блока.

В исходном состоянии все ключи матричной клавиатуры К разомкнуты, V=0, процессор пассивен, светодиод не излучает энергию. Благодаря наличию резисторов DB0=DB1=...=DB7=1 (напряжения

высокого уровня), P10=P11=...=P17=1, группа инверторов формирует на выходах сигналы лог. 0, которые передаются на строки матричной клавиатуры К. Триггер D3 — D4 установлен в состояние A=1, в результате на вход R триггера D2 постоянно действует сигнал установки в 0, нулевой сигнал с выхода этого триггера воздействует на вход R начальной установки процессора. На выходе одновибратора F сформирован сигнал лог. 1, на входе S установки 1 триггера D1 — сигнал лог. 0, на выходах ALE и P27 процессора — сигналы лог. 1.

При нажатии на любую клавишу напряжение низкого уровня (около 1 В) с анода соответствующего диода поступает в столбец матрицы, в результате на выходе элемента И — НЕ D3 формируется сигнал лог. 1, триггер D3 — D4 переходит в состояние A=0. Сигнал установки 0 снимается в входа R триггера D2, однако его исходное состояние не меняется. Триггер D1 устанавливается в 1 по входу S, процессор переходит в активное состояние (V=1), но поддерживается в режиме начальной установки, так как триггер D2 по-прежнему находится в состоянии лог. 0. На выходе одновибратора F формируется отрицательный импульс длительностью 16 мс.

По окончании импульса на выходе одновибратора F (при переходе сигнала из 0 в 1) триггер D2 устанавливается в 1 по входу C, в результате процессор приступает к выполнению программы обслуживания клавиатуры. При работе процессора на его выходе ALE формируются импульсы, подтверждающие истинность информации на адресной шине (не показана). Эти импульсы поступают на вход синхронизации триггера D1, но не воздействуют на этот триггер, так как он постоянно поддерживается в состоянии 1 сигналом лог. 1 на входе S.

Программа опроса клавиатуры предусматривает выдачу «бегущего 0» на выходы P10 — P17 процессора и считывание информации, поступающей при этом на входы DB0 — DB7. При опросе на все строки матричной клавиатуры, за исключением одной, подаются сигналы лог. 0. Если замкнутый ключ находится в выбранной строке, то напряжение на всех столбцах соответствует лог. 1; если выбрана любая другая строка, то в столбце, к которому подключен замкнутый ключ, устанавливается лог. 0. Предполагается, что в каждом столбце не может быть двух или более одновременно замкнутых ключей. Такие ситуации выявляются программно и рассматриваются как ошибки оператора; при этом диоды обеспечивают развязку выходов инверторов, чтобы не было электрических перегрузок. «Дребезг» контактов клавиатуры компенсируется программно, с помощью проведения нескольких серий опросов.

При завершении ввода команды, соответствующей нажатию одной или (последовательно) нескольких клавиш, процессор формирует на выходе P25 нужную последовательность импульсов, которая проходит через усилитель и передается в виде импульсов инфракрасного излучения в сторону устройства-приемника. По окончании передачи (или при обнаружении различного рода некорректных действий оператора) процессор выдает на выход P27 отрицательный импульс, который приводит к безусловному формированию сигнала A=1. Триггер D2 устанавливается в 0, сигнал установки 1 триггера D1 снимается, так что по фронту сигнала ALE триггер D1 также переходит в состояние 0, процессор возвращается в режим пассивного ожидания следующего сеанса связи.

5.31. Схема связи микроЭВМ с удаленными абонентами [172]

Схема рис. 5.71 содержит приемопередатчик, соединенный с одним разрядом выходного и одним разрядом входного порта микроЭВМ, и двухпроводную магистраль М с подключенными к ней абонентами. Абонент включает контроллер К периферийного устройства (цифро-аналогового преобразователя, дисплея, печатающего устройства и т. п.), входную и выходную оптронные цепи. Абоненты гальванически развязаны между собой и с приемопередатчиком.

Схема обеспечивает двусторонний обмен данными между микроЭВМ и выбранным абонентом. Выбор абонента производится посылкой из микроЭВМ в магистраль М адресного кода, который

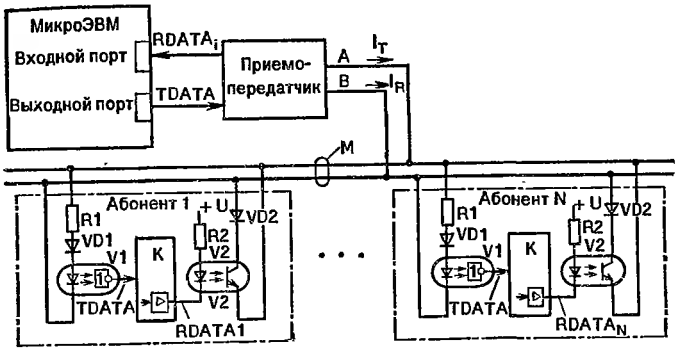


Рис. 5.71. Схема связи микроЭВМ с удаленными абонентами

распознаётся этим абонентом. После передачи адресного кода из микроЭВМ передаются коды, определяющие тип предстоящей операции (чтение, запись, чтение — модификация — запись и т. п.) и другие параметры, например указывающие способ кодирования сообщения. Непосредственное взаимодействие абонентов по магистрали М не предусмотрено.

Входные цепи оптронов V1 и выходные цепи оптронов V2 абонентов активизируются магистральными напряжениями разной полярности. Чтобы вызвать ток через входные цепи оптронов V1, необходимо чтобы напряжение между проводами А и В магистрали М было «положительным» — потенциал провода А должен быть выше потенциала провода В. Если полярность напряжения между проводами А и В магистрали М обратная, то ток через входные цепи оптронов V1 не течет, но при этом транзисторы оптронов V2 способны реагировать на свет, излучаемый соответствующими светодиодами. Логика работы схемы такова, что информация передается через оптрон V2 только из выбранного абонента; остальные (невыбранные) абоненты пассивны, их оптроны V2 выключены. Рассмотрим работу схемы в режиме передачи информации к абоненту и от абонента.

При передаче адреса или данных микроЭВМ формирует после-

довательность сигналов TDATA, используя, например «телеграфный» 7-битовый код, сопровождаемый «стартовым» и «стоповым» битами (см. гл. 3). Выдаваемые сигналы проходят через приемопередатчик и преобразуются им в знакопеременное напряжение между проводами А и В магистрали М. Если TDATA=0 (напряжение низкого уровня), то напряжение между проводами А и В положительно (+А, —В), если TDATA=1, то это напряжение отрицательно (—А, +В). При положительном напряжении ток $I_т$ разветвляется между входными цепями оптронов V1, в результате светодиоды этих оптронов излучают свет и на входы всех контроллеров К поступают сигналы лог. 0. При отрицательном напряжении входные цепи оптронов V1 закрыты, поэтому на входы всех контроллеров К поступают сигналы лог. 1. Диод VD1 предохраняет светодиод оптрона от необратимого пробоя при воздействии обратного напряжения. При передаче информации от микроЭВМ к абонентам (абоненту) все оптроны V2 должны быть выключены сигналами $RDATA_i=1$ из контроллеров К.

После выбора абонента и передачи ему кода команды чтения (и, возможно, других параметров) микроЭВМ на время приема данных устанавливает сигнал TDATA=1, приемопередатчик формирует в магистрали М отрицательное напряжение (—А, +В). Выбранный (i-й) абонент формирует последовательность сигналов $RDATA_i$, которые проходят через оптрон V2, магистраль М, приемопередатчик и поступают во входной порт микроЭВМ.

Если выбранный контроллер К сформировал сигнал $RDATA_i=0$, то светодиод оптрона V2 излучает свет, транзистор этого оптрона насыщен, в магистрали М протекает «отрицательный» ток I_R, который регистрируется приемопередатчиком, в результате во входной порт микроЭВМ поступает сигнал $RDATA_i=0$. Если выбранный контроллер К сформировал сигнал $RDATA_i=1$, то оптрон V2 выключен, ток I_R практически отсутствует, во входной порт микроЭВМ поступает сигнал $RDATA_i=1$. Диод VD2 предотвращает протекание тока утечки через транзистор оптрона V2 при подаче на него обратного напряжения. Если этот ток мал, то диод VD2 можно исключить из схемы.

Приемопередатчик (рис. 5.72) содержит шесть инверторов с открытым коллектором, два двухтактных выходных каскада на транзисторах VT2—VT3 и VT4—VT5 и детектор тока I_R на транзисторе VT1.

Если TDATA=0, то транзисторы VT2 и VT5 включены, транзисторы VT3 и VT4 выключены. В результате в провод А подается положительное напряжение, близкое напряжению +U, а в провод В — напряжение, близкое нулю. Через входные цепи оптронов V1 всех абонентов протекает ток, как было описано ранее. Сигнал $RDATA_i=1$, так как транзистор VT1 закрыт.

Если TDATA=1, то транзисторы VT2 и VT5 выключены, транзисторы VT3 и VT4 включены. В провод В подается положительное напряжение высокого уровня. В зависимости от состояния абонента провода А и В либо разомкнуты (при передаче или приеме лог. 1), либо замкнуты через насыщенный транзистор абонентского оптрона V2 и диод VD2 (при приеме в микроЭВМ лог. 0). В первой ситуации ток I_R практически отсутствует, транзистор VT1 выключен, $RDATA_i=1$, во второй — ток I_R достаточен для насыщения транзистора VT1, в результате $RDATA_i=0$.

Если абоненты находятся на небольшом расстоянии от микро-

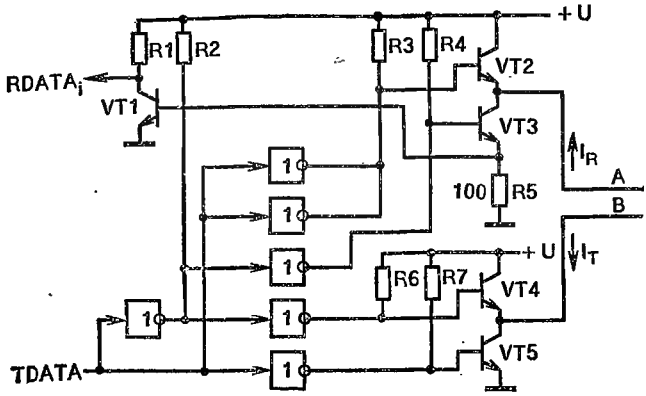


Рис. 5.72. Схема приемопередатчика (см. рис. 5.71)

ЭВМ, то можно вместо входной оптронной цепи (R1 — VD1 — V1) использовать компаратор напряжений, а вместо выходной цепи (VD2 — V2) — дифференциальный усилитель с парафазным выходом.

5.32. Связь микроЭВМ с клавиатурой с использованием цифро-аналогового преобразователя [175]

Схема рис. 5.73 содержит клавиатуру (ключи SW1 — SW7), компаратор К напряжений, микроЭВМ, цифро-аналоговый преобразователь ЦАП и группу резисторов.

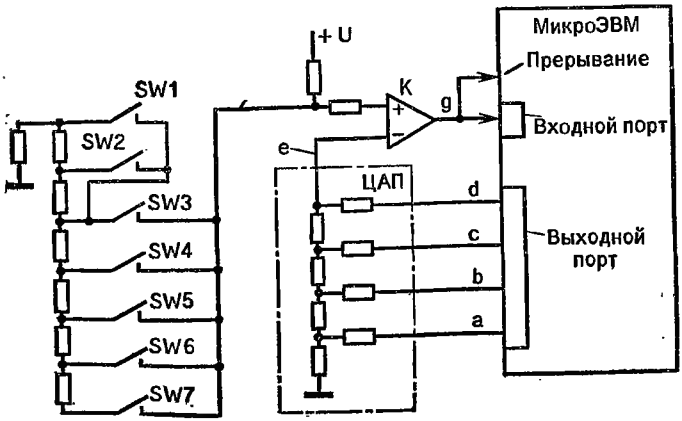


Рис. 5.73. Схема сопряжения микроЭВМ с клавиатурой

В исходном состоянии микроЭВМ работает по некоторой программе, не связанной с опросом клавиатуры, ключи SW3 — SW7 разомкнуты, ключи SW1 и SW2 — в произвольных положениях. В выходном порте микроЭВМ установлен код 1111 (напряжения высокого уровня), так что в точке e присутствует максимальное напряжение, меньшее, чем в точке f. На выходе компаратора K присутствует сигнал лог. 1.

При замыкании одного из ключей SW3 — SW7 напряжение в точке f снижается до некоторого уровня, меньшего, чем напряжение в точке e. На выходе компаратора K формируется серия сигналов лог. 0, а затем постоянный сигнал лог. 0, так как в момент замыкания ключа обычно наблюдается «дребезг» контактов. Фронт первого сигнала лог. 0 вызывает прерывание микроЭВМ. Процессор временно откладывает текущее задание, запоминает параметры возврата к нему и переходит к прерывающей программе.

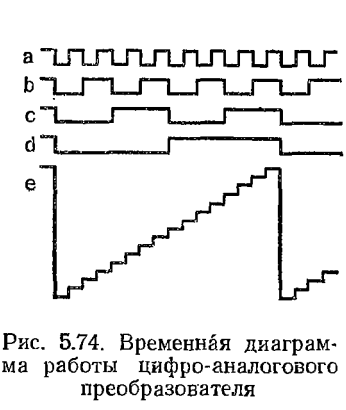


Рис. 5.74. Временна́я диаграмма работы цифро-аналогового преобразователя

В этой программе предусмотрен запрет реакции на новые сигналы прерывания от клавиатуры (из-за дребезга контактов и выполнения описанной далее процедуры поиска) и введена задержка, компенсирующая время «успокоения» контактов ключа. Чтобы определить, какой ключ замкнут, микроЭВМ формирует в выходном порте последовательность сигналов в соответствии с временно́й диаграммой, показанной на рис. 5.74, и следит за сигналом, поступающим в ее входной порт. В моменты перехода сигнала g из 1 в 0 запоминается код, выданный в выходной порт. После проведения ряда таких экспериментов и получения устойчивого результата (кода в выходном порте, при котором наблюдается срабатывание компаратора) по коду определяется ситуация на клавиатуре. После этого процессор устанавливает в выходном порте исходный код 1111, разрешает реакцию на отрицательный фронт сигнала прерывания и возвращается к прерванному заданию.

Размыкание ключа также сопровождается дребезгом контактов, поэтому микроЭВМ получает сигнал прерывания и в этом случае, хотя он и не несет полезной информации (следовало бы реагировать только на замыкание ключа). МикроЭВМ выполняет описанные ранее действия, но на этот раз она определяет, что в установившемся режиме напряжение в точке f более высокое, чем в точке e, т. е. ключи SW3 — SW7 разомкнуты.

Ключи SW1 и SW2 задают режимы работы клавиатуры, шунтируя два или один резистор делителя напряжения. В зависимости от их состояния микроЭВМ по-разному воспринимает замыкания исполнительных ключей SW3 — SW7.

Выход компаратора K можно подключать не к двум, а к одному входу микроЭВМ. Если использовать только разряд входного порта, то микроЭВМ должна периодически опрашивать сигнал в этом разряде, например при прерываниях от сетевого таймера частотой

50 Гц. При обнаружении во входном порте сигнала лог. 0 определяется ситуация на клавиатуре, как было описано ранее. Если использовать только вход прерывания, то придется усложнить прерывающую программу и предусмотреть режим вложенных прерываний с достаточно большим числом уровней вложения (когда прерывающая программа прерывается этой же программой, а эта, в свою очередь, также прерывается и т. д.).

5.33. Уменьшение аппаратных затрат при задании собственных номеров однотипных устройств [181]

В § 5.5 приведено возможное решение задачи минимизации числа внешних контактов при автоматическом задании собственных номеров однотипных устройств. Однако при наличии достаточного числа свободных контактов на платах можно уменьшить аппаратные затраты, соединив платы, как показано на рис. 5.75. В табл. 5.6 при-

Таблица 5.6

Позиция платы	Код номера платы						
	A	B	C	D	E	F	G
1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	0
3	1	1	1	1	1	0	0
4	1	1	1	1	0	0	0
5	1	1	1	0	0	0	0
6	1	1	0	0	0	0	0
7	1	0	0	0	0	0	0
8	0	0	0	0	0	0	0

ведены коды собственных номеров, подаваемые на однотипные платы в зависимости от их позиций в ряду. В первой плате формируется код 1111111, во второй — код 1111110 и т. д.

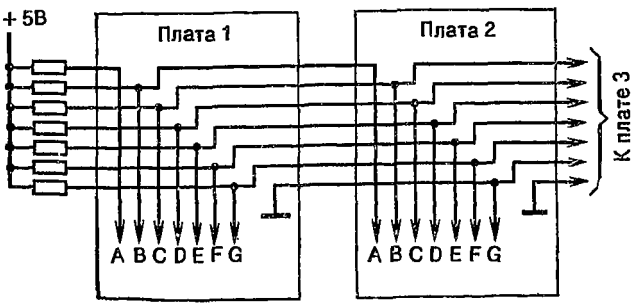


Рис. 5.75. Схема автоматического задания собственного номера платы

Можно преобразовать код A — G в двоичный с использованием шифратора, если это необходимо. Шину +5 В можно заменить шиной 0 В и замкнуть накоротко резисторы; при этом в каждой плате нижний правый контакт отключается от шины 0 В и через резистор соединяется с шиной +5 В. Приведенная схема позволяет исключить из платы микропереключатели, перемычки и т. п. элементы, с помощью которых обычно задают ее собственный номер, чтобы все платы отличались друг от друга с «точки зрения» центрального процессора.

5.34. Обработка прерываний в многопроцессорной системе [190]

В многопроцессорной системе обработку внешних запросов на прерывание можно поручить одному процессору или распределить ее между несколькими процессорами. В системе, показанной на рис. 5.76, имеются четыре процессора, размещенных на разных платах. Три сигнала запросов на прерывание обрабатываются одним и тем

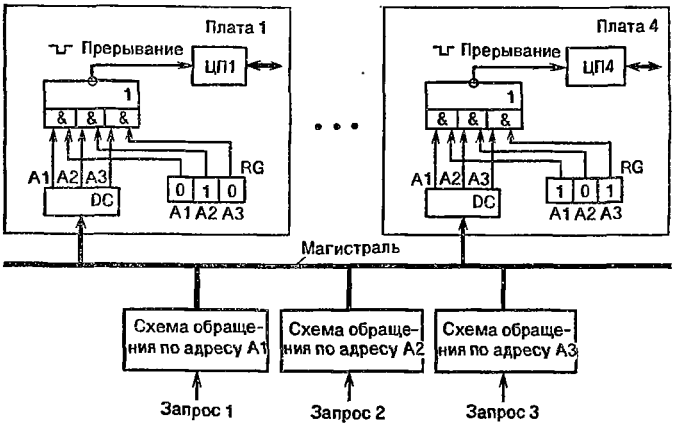


Рис. 5.76. Многопроцессорная система с программным назначением процессоров, обслуживающих источники запросов на прерывание

же или разными процессорами в зависимости от кодов, записанных в регистры RG. Поступающие от внешних устройств запросы «превращаются» в адресные обращения по магистрали, а затем — в сигналы прерываний процессоров, назначенных системой для обработки соответствующих запросов. Это позволяет упростить структуру магистрали исключением из нее линий для передачи запросных сигналов.

Каждому запросу соответствует одна ячейка свободной области адресного пространства общей магистрали. (При использовании ме-

тода частичной дешифрации каждая ячейка может иметь в адресном пространстве ряд копий, см. гл. 1.) Запросам 1, 2 и 3 в данном примере соответствуют ячейки с адресами А1, А2 и А3 (это не шестнадцатеричная запись, а условные «имена» ячеек). Дешифраторы DC реагируют на появление этих адресов в магистрали и формируют на соответствующих выходах сигналы лог. 1. При совпадении сигналов лог. 1 на двух входах И элемента И — ИЛИ — НЕ на его выходе формируется сигнал лог. 0, вызывающий прерывание процессора.

В данном примере загрузка регистров RG такова, что процессор ЦП1 реагирует на сигнал Запрос 2, а процессор ЦП4 — на сигналы Запрос 1 или Запрос 3. Процессоры ЦП2 и ЦП3, как предполагается, нечувствительны к сигналам Запрос 1—3, так как в соответствующих программно-доступных регистрах RG (не показаны) записаны нулевые коды. (Отметим, что возможна и одновременная обработка запроса несколькими процессорами, например для повышения надежности обработки.)

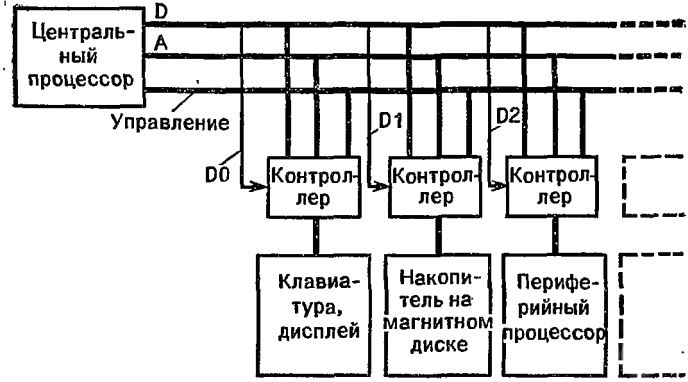
При поступлении сигнала Запрос 1 схема обращения по адресу А1 с разрешения арбитра (не показан) захватывает магистраль и выполняет цикл записи или чтения информации по этому адресу. В приведенном примере шина данных магистрали не используется. Адрес А1 распознаётся одновременно всеми дешифраторами DC, однако сигнал прерывания посылается только в процессор ЦП4, так как в левом разряде соответствующего регистра RG записана лог. 1. Процессор ЦП4 приступает к выполнению прерывающей программы, а остальные процессоры продолжают выполнение фоновых заданий. Запросы 2 и 3 обрабатываются аналогично.

Можно несколько усовершенствовать схему рис. 5.76, чтобы уменьшить число занятых адресов до одного (А*). В этом случае схемы обращения к магистрали по сигналам запросов и с разрешения арбитра выполняют циклы записи по одному и тому же адресу А* с выдачей лог. 1 в одну из линий шины данных (каждая схема подключена к своей линии). При опознании адреса А* код, выданный в шину данных, сравнивается в каждой плате с кодом, записанным в регистре RG; точнее говоря, анализируется возможность обработки текущего запроса каждым процессором, как и в рассмотренной схеме. В результате определяется процессор (процессоры), которому адресовано прерывание. Для «уплотнения» информации в шине данных можно использовать шифратор. При этом вместо кода, содержащего единицу в одном разряде шины данных, могут формироваться любые коды, в том числе и нулевой, однако усложняются схемы их анализа при проверке альтернативных условий, как например, в плате 4.

5.35. Программно-управляемое назначение собственных адресов контроллеров [191]

Для задания собственных адресов контроллеров обычно используют микропереключатели или паяные перемычки на платах. Это создает неудобства в работе, приводит к ошибкам при назначении адресов, снижает степень интеграции аппаратуры контроллеров. Предлагаются два варианта построения систем с программно-управляемым назначением собственных адресов контроллеров, рис. 5.77.

На этапе задания собственного адреса контроллера (рис. 5.78)



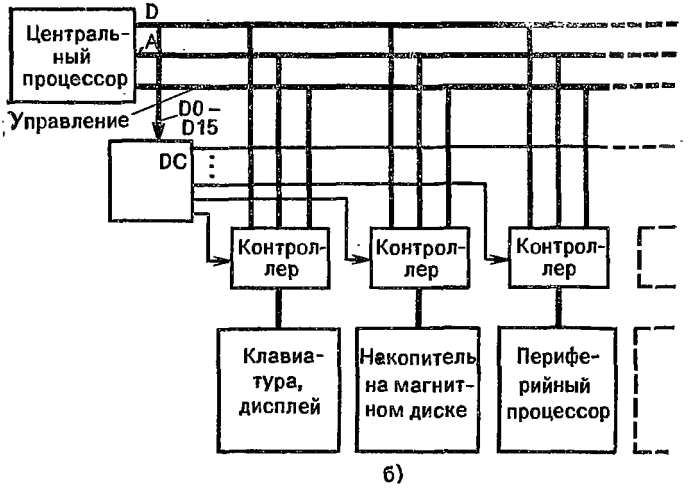


Рис. 5.77. Система с программно-управляемым назначением собственных адресов контроллеров.

При задании собственных адресов для выбора контроллера используются разные линии шины данных (*а*) или выходы дешифратора DC (*б*), что позволяет увеличить число контроллеров

центральный процессор выполняет операцию записи по некоторому адресу. Старшие 8 разрядов этого адреса (A8 — A15) распознаются дешифраторами DC всех контроллеров, в результате двухвходовые элементы И этих контроллеров открываются по верхним входам. Код, выдаваемый центральным процессором при записи в шину данных D

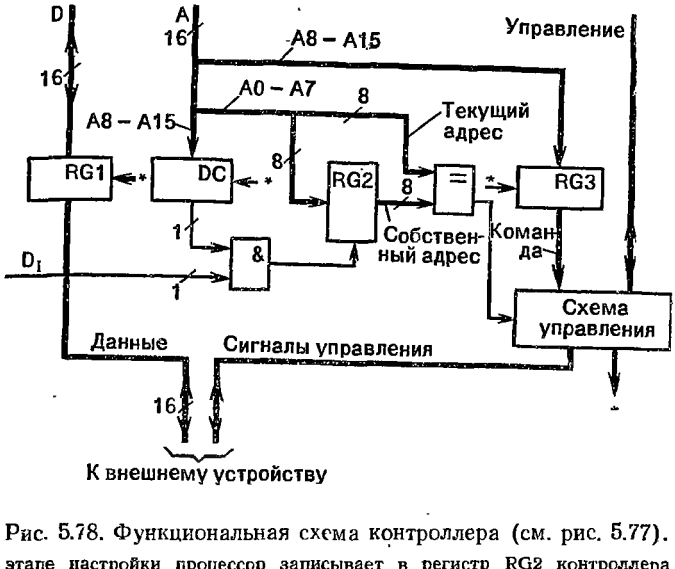


Рис. 5.78. Функциональная схема контроллера (см. рис. 5.77).

На этапе настройки процессор записывает в регистр RG2 контроллера его собственный адрес

(рис. 5.77, *а*), содержит единицу в разряде D_1, который поступает на нижний вход элемента И выбранного контроллера, в остальных разрядах этого кода присутствуют нули. В результате в выбранном контроллере срабатывает элемент И, в регистр RG2 принимаются младшие разряды адреса, в которых процессор указал собственный адрес данного контроллера. Остальные контроллеры остаются в пассивном состоянии, так как их элементы И закрыты по нижним входам. Такая процедура повторяется для всех контроллеров, которым присваиваются различные адреса. Число контроллеров в схеме рис. 5.77, *а* не может превышать разрядности шины данных.

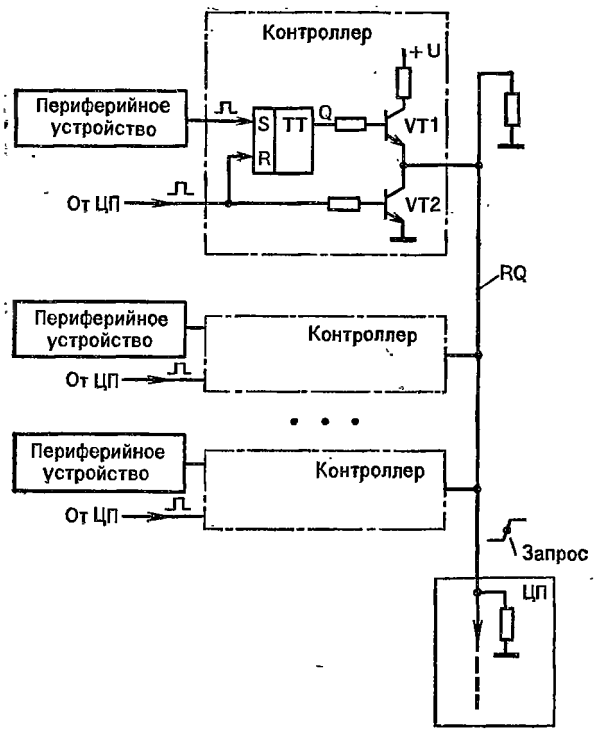
В дальнейшем, при поступлении по шине управления признака обращения к устройствам ввода-вывода и совпадении текущего адреса с собственным, схема управления контроллера вырабатывает серию управляющих сигналов (внутренние сигналы контроллера помечены знаками *), которые необходимы для выполнения команды, код которой принят в регистр RG3. В процессе выполнения команды данные принимаются с шины данных D в регистр RG1 или выдаются в эту шину.

В схеме рис. 5.77, *б* для увеличения числа контроллеров использован дешифратор DC. Для выбора контроллеров можно задать любые кодовые комбинации из диапазона 0000 — FFFF, которые распознаются дешифратором. Такая схема может оказаться полезной, когда разрядность шины данных составляет 8 (а не 16 бит,

5.36. Формирование импульсных сигналов запросов прерываний в потенциальной линии связи типа Монтажное ИЛИ [202]

Если в микроЭВМ имеется несколько контроллеров, то объединяющая их линия запроса на прерывание обычно выполняет функцию Монтажное ИЛИ (И) и служит для получения обобщенного запросного сигнала, который передается в центральный процессор. Однако если процессор реагирует не на уровень напряжения, а на фронт сигнала (переход из 0 в 1, как в микроЭВМ IBM PC), то при суммировании в линии по схеме ИЛИ потенциальных запросных сигналов, перекрывающихся во времени, фронты этих сигналов теряются и, следовательно, система «насыщается» запросами, а процессор не может их воспринимать.

В схеме рис. 5.79, *а* этот недостаток устранен — после обработки очередного запроса и при наличии одного или более конкурирующих запросов в линии RQ принудительно устанавливается сигнал лог. 0 (напряжение низкого уровня), который затем снимается. В резуль-



а)

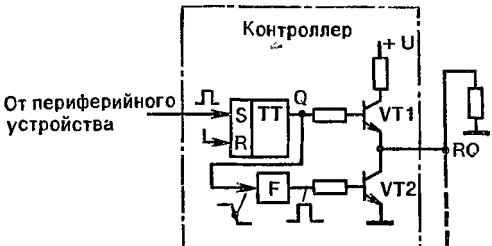


Рис. 5.79. Система связи контроллеров с центральным процессором (*а*); вариант схемы контроллера с автономным формирователем F импульса временно́го разделения запросов (*б*)

тате искусственно формируется фронт сигнала, воспринимаемый процессором ЦП. Рассмотрим работу схемы более подробно.

В исходном состоянии центральный процессор выполняет фоновую программу, не связанную с обслуживанием контроллеров. Запросные триггеры ТТ всех контроллеров установлены в 0, сигналы установки 0 триггеров ТТ отсутствуют (равны лог. 0), в линии RQ за счет резисторов, подключенных к шине нулевого потенциала, поддерживается сигнал лог. 0. При получении сигнала (положительного импульса) от периферийного устройства триггер ТТ контроллера устанавливается в 1, в результате открывается транзистор VT1, в линии RQ формируется сигнал лог. 1, фронт которого воздействует на процессор.

Процессор переходит в режим прерывания, обслуживает контроллер (методы поиска источника прерывания рассмотрены в § 2.5), и в процессе или после обслуживания устанавливает в 0 триггер ТТ выбранного контроллера подачей на его вход R положительного импульса.

Этот импульс временно открывает транзистор VT2, поэтому в линии RQ формируется сигнал лог. 0. Если конкурирующих запросов нет, то сигнал лог. 0 продолжает присутствовать в линии RQ и по окончании импульса сброса триггера ТТ. Если имеются запросы от других контроллеров, то по окончании импульса сброса триггера ТТ сигнал лог. 1 восстанавливается в линии RQ, так как в контроллерах-конкурентах транзисторы VT1 открыты. Таким образом, формируется фронт сигнала на входе запроса прерывания центрального процессора.

При отсутствии возможности установки 0 триггера ТТ сигналом от процессора (когда этот триггер устанавливается в 0 самим контроллером) можно использовать формирователь F импульса, который срабатывает в момент перехода триггера ТТ из 1 в 0, рис. 79, *б*,

Глава 6

Контроль, диагностика, восстановление, защита и поиск информации

6.1. Метод следящего самоконтроля микроЭВМ на основе предварительного прогнозирования вариантов ее «поведения» [53]

При использовании микроЭВМ в системе, работающей в реальном масштабе времени, не исключена возможность того, что в результате сбоя или отказа она начнет беспорядочное «блуждание» по памяти, рассматривая числовые массивы как цепочки команд, команды — как данные или адреса и т. д. Поскольку неправильное управление недопустимо для многих объектов, возникает задача быстрого обнаружения подобных «срывов».

Любую программу, записанную в машинных кодах в память микроЭВМ, можно представить в виде конечного числа линейных участков, связанных между собой командами переходов (рис. 6.1, *а*). В отсутствие внешних прерываний «поведение» микроЭВМ на каждом линейном участке строго детерминировано в том смысле, что при правильном его прохождении на управляющих линиях общей

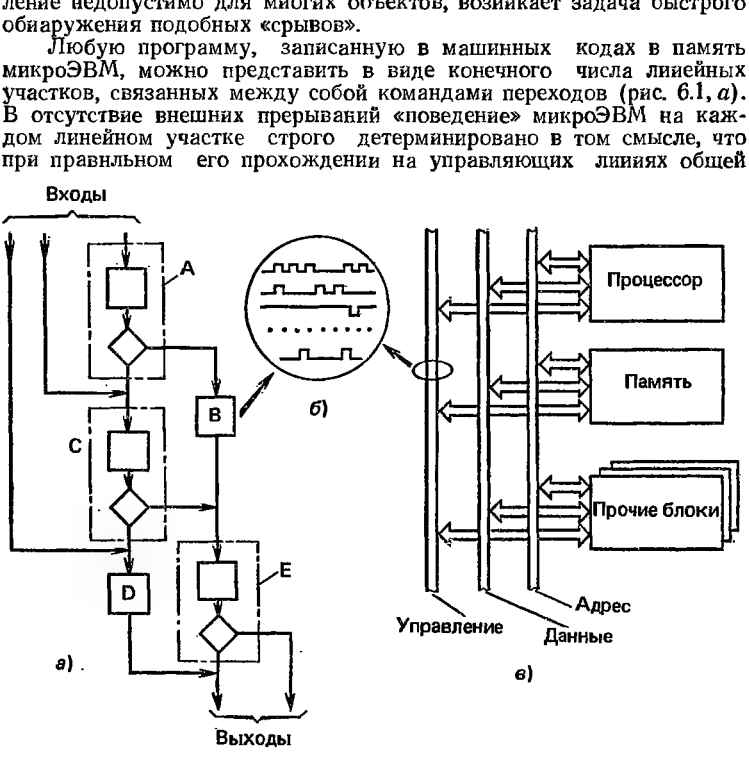


Рис. 6.1. Предсказание «поведения» микроЭВМ при работе по произвольной программе:

а — фрагмент программы; *б* — «поведение» микроЭВМ на линейном участке B; *в* — структурная схема микроЭВМ

магистрали будет сформирована строго определенная временная диаграмма, которую можно «вычислить» еще до начала прогона самой программы (рис. 6.1, *б*).

В простейшем случае подсчитывается ожидаемое число импульсов на каждой управляющей линии. Вычисления сводятся к суммированию констант, определяющих число импульсов на выбранной управляющей линии при выполнении каждой команды данного линейного участка. В результате предварительных подсчетов выясняется, например, что при правильном прохождении участка А программы на первой управляющей линии должно быть зарегистрировано 27 импульсов, на второй — 19 и т. д.

Предполагается также, что совокупность управляющих сигналов, которыми сопровождается выполнение команд условных переходов, не зависит от того, выполнено проверяемое условие или нет. Это предположение является общепринятым для многих микроЭВМ и в данном методе позволяет рассматривать команду условного перехода как завершающую команду линейного участка (см. участок А на рис. 6.1, *а*). Процесс прогнозирования поддается автоматизации, поскольку выделение в произвольной программе всех линейных участков производится путем анализа адресной информации, содержащейся в программе, без прогона самой программы. Необходимо знать лишь адреса внешних входов в эту программу. В результате прогнозирования формируется массив вспомогательной информации о программе. Формирование этого массива может производиться как на кроссистеме программирования, т. е. «вдали» от исполнительной микроЭВМ, так и на самой исполнительной микроЭВМ (после трансляции программы).

Рис. 6.2. Структурная схема микроЭВМ с системой самоконтроля

При работе системы самоконтроля (рис. 6.2) одновременно с выбором из основной памяти начальной команды некоторого линейного участка из дополнительной памяти (по отдельным шинам) извлекается служебная информация, которая помещается в аппаратные счетчики, содержимое их уменьшается по мере приближения к концу данного линейного участка. (Разрядность счетчиков может быть небольшой; в этом случае ведется счет по модулю.) При выходе на начало другого линейного участка счетчики проверяются на равенство нулю их содержимого, затем они вновь загружаются и т. д. Таким образом, ход выполнения любой, даже сильно разветвленной, программы постоянно контролируется. «Срыв» не будет зарегистрирован только в том случае, если он произошел точно в конце какого-либо линейного участка и микроЭВМ «ушла» точно

на начало «чужого» линейного участка. Вероятность именно таких «точных» срывов мала. Время между возникновением сбоя или отказа и его обнаружением не превышает времени прохождения линейного участка максимальной длины и, поскольку последний может быть искусственно разбит на достаточно короткие звенья, может составлять, например, 100 мкс.

При обработке прерываний в число параметров возврата, «упрятываемых» в стек, входит содержимое счетчиков, которое восстанавливается при возврате с учетом необходимой поправки на «лишние» импульсы, сформированные на управляющих линиях в связи с выполнением стандартных процедур упрятывания — восстановления.

Отметим, что пользователь «не подозревает» о существовании такого механизма самоконтроля, так как вся необходимая информация генерируется без его участия.

6.2. Устройство для контроля работы микроЭВМ путем анализа времени выполнения программного цикла [54]

Если рабочая программа выполняется циклически с известным временем цикла (например, лежащим в пределах от 5,5 до 6,2 мс), то, проверяя в каждом цикле время ее выполнения, можно оперативно следить за работой микроЭВМ (рис. 6.3).

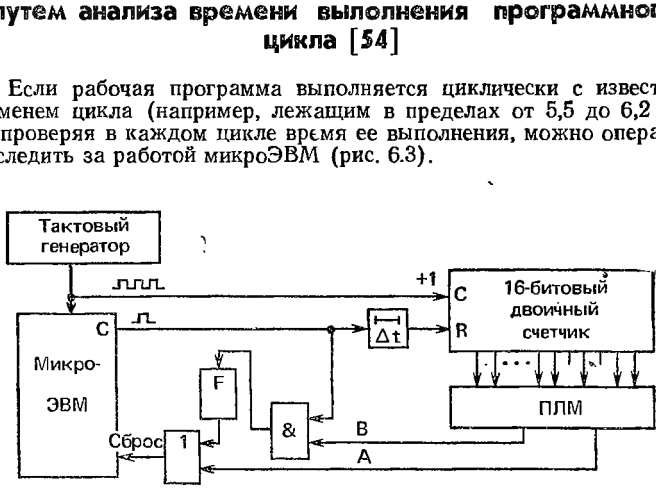


Рис. 6.3. Схема контроля микроЭВМ

Сигнал в точке А принимает единичное значение, если в счетчике зафиксировано время, превышающее максимальное время выполнения одного цикла программы (в нашем примере — 6,2 мс). Сигнал в точке В принимает единичное значение, если $t<5{,}5$ мс. Сигналы А и В в данном примере вырабатываются с помощью программируемой логической матрицы ПЛМ (см. § 1.4) с соответствующей кодировкой.

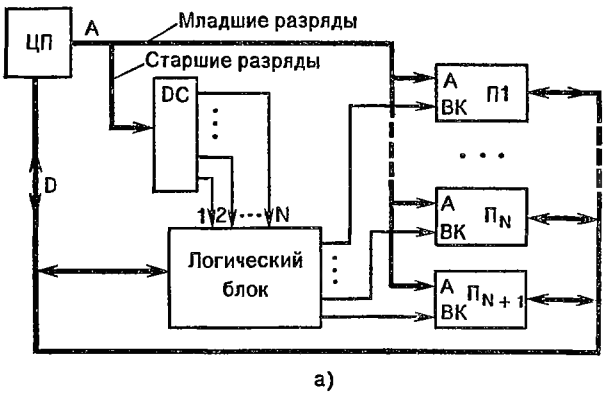
В конце каждого цикла программы микроЭВМ генерирует импульс опроса на выходе С. Если время лежит в заданных пределах, то в момент поступления импульса сигнал В=0 и элемент И не срабатывает. Если сигнал С возник слишком рано (программа «провалилась» на выход, минуя какие-то участки), то в момент формирования импульса С сигнал В=1, в результате срабатывает цепь эле-

мент И — одновибратор F—элемент ИЛИ и на вход сброса микроЭВМ поступает импульс, возвращающий ее в исходное состояние, с попыткой вновь войти в рабочий режим. То же происходит и при превышении заданного времени (A=1).

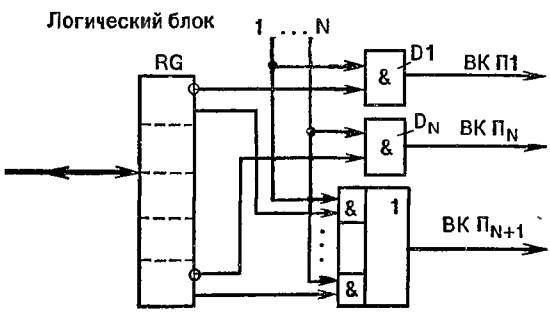
Элемент задержки предотвращает возможные «гонки» — иначе может получиться так, что сигнал B успеет принять единичное значение прежде, чем кончится импульс C при нормальной работе. С другой стороны, задержка не должна быть слишком большой, иначе при нормальной работе может ошибочно возникнуть сигнал A.

6.3. Программно-управляемая замена одного из блоков памяти на резервный блок [55]

На рис. 6.4, *а* показана микроЭВМ, содержащая центральный процессор ЦП, рабочие блоки памяти П1—$П_N$, резервный блок памяти $П_{N+1}$, дешифратор DC и логический блок, схема которого приведена на рис. 6.4, *б*.



а)



б)

Рис. 6.4. Система с резервным блоком памяти $П_{N+1}$ (*а*) и структура логического блока (*б*)

При нормальной работе микроЭВМ в программно-доступный регистр RG записан нулевой код, поэтому элементы И ($D1—D_N$) открыты по нижним входам, элемент И—ИЛИ закрыт. Поэтому сигналы с выходов дешифратора проходят на входы выбора кристалла ВК соответствующих блоков памяти. При обнаружении отказа одного из блоков (механизм обнаружения может быть любым) процессор записывает единицу в соответствующий разряд регистра RG. При этом закрывается соответствующий элемент И из группы $D1—D_N$ и в случае обращения к неисправному блоку сработает элемент И—ИЛИ, открывая блок $П_{N+1}$ для доступа. Информация в отказавшем блоке теряется, поэтому после переключения блоков решение задачи повторяется либо с самого начала, либо с некоторой «опорной» точки.

6.4. Однокристальная (или одноплатная) микроЭВМ, «создающая себя» при включении напряжения питания из имеющихся исправных блоков [56]

МикроЭВМ (рис. 6.5) содержит несколько одинаковых процессоров ЦП, ПЗУ и ОЗУ. Процессор ЦП включает «собственно процессор» ЦП* и схему распространения стартового сигнала $S_{вх}$ (рис. 6.6). При подаче стартового сигнала $S_{вх}$ триггер D1 устанавливается

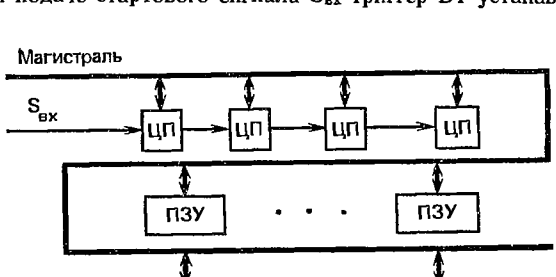


Рис. 6.5. Резервированная микроЭВМ

в 1 и ЦП* начинает процедуру тестирования. Условие E=1 соответствует подключению данного процессора к общей магистрали. Если за приемлемое время получен положительный результат тестирования, то сигнал Финиш равен 1, триггер D2 устанавливается в 1, элемент И закрывается, так что сигнал с выхода элемента задержки не сможет пройти к следующему процессору. Если же тестирование не завершилось успехом к моменту прохождения сигнала $S_{вх}$ через элемент задержки, то срабатывает элемент И, триггер D1 устанавливается в 0, отключая данный процессор от магистрали, а стартовый сигнал передается в следующий процессор.

Порядок тестирования.

1. При включении напряжения питания микроЭВМ вырабатыва-

ется сигнал $S_{вх}$, поступающий на первый процессор. Этот процессор считывает содержимое первого ПЗУ и сравнивает полученную контрольную сумму с кодом, записанным в последней ячейке этого ПЗУ (действия по п. 1 выполняются под управлением микропрограммы, хранящейся в процессоре). Если коды не совпали, процессор переходит к проверке следующего ПЗУ и т. д. Кодировка всех ПЗУ одинакова. Если работающее ПЗУ не найдено данным процессором, то запускается следующий процессор.

2. Когда работающее ПЗУ найдено, процессор, работая по программе, хранящейся в этом ПЗУ, проверяет себя. Если тестирование

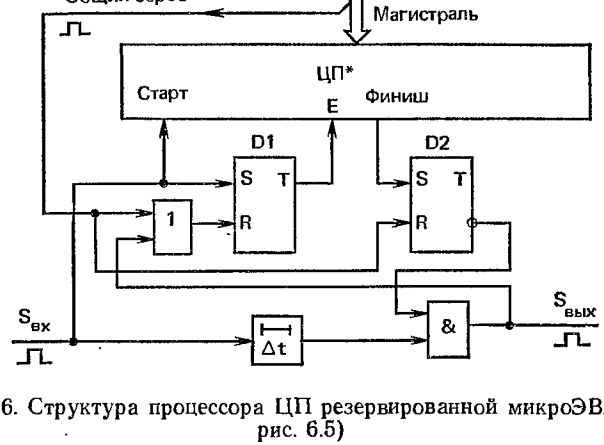


Рис. 6.6. Структура процессора ЦП резервированной микроЭВМ (см. рис. 6.5)

процессора произведено удачно, то вырабатывается единичный сигнал Финиш и распространение стартового сигнала запрещается. В противном случае данный процессор отключается и стартовый сигнал поступает на следующий процессор. Это процесс продолжается до тех пор, пока не будет найдена работающая пара процессор — ПЗУ. Если этого не произошло, то микроЭВМ признается неисправной (в течение, например, 10 с на экране дисплея, подключенного к микроЭВМ, не появилось сообщение о готовности микроЭВМ к работе).

3. После того как найдена пара процессор — ПЗУ, производится тестирование блоков ОЗУ с запоминанием таблицы годности либо на регистрах процессора, либо в первом обнаруженном исправном ОЗУ. Тест ОЗУ хранится в ПЗУ.

В микроЭВМ используется принцип выделения страниц ОЗУ по требованию пользовательских программ. Каждой странице соответствует отдельный блок ОЗУ. Программам выделяются только исправные страницы, а неисправные рассматриваются операционной системой как недоступные для обращения. В микроЭВМ могут быть введены и другие устройства, например контроллеры, представленные в «множественном числе».

Таким образом, всякий раз при включении напряжения питания

микроЭВМ «собирает себя» из отдельных блоков. Достаточным условием обеспечения работоспособности является наличие единичных исправных экземпляров всех входящих в нее блоков и исправность общих коммуникационных элементов.

6.5. Коррекция постоянных ошибок в ОЗУ методом двойной инверсии [57]

Процесс коррекции отражен в табл. 6.1.

Таблица 6.1

Последовательность коррекции	Номер разряда 0 1 2 3 4 *5 6 7	$\overline{\text{mod 2}}$	Примечание
Запись 1	1 1 0 1 0 0 1 1	0	Исходное слово данных
Чтение 1	1 1 0 1 0 1 1 1	0	Ошибка при проверке на нечетность
$D \to \overline{D}$	0 0 1 0 1 0 0 0	1	Данные инвертируются
Запись 2	0 0 1 0 1 0 0 0	1	Запись инверсных данных
Чтение 2	0 0 1 0 1 1 0 0	1	Ошибка при проверке на нечетность
$\overline{D} \to D$	1 1 0 1 0 0 1 1	0	Данные инвертируются, ошибка исправлена

Предположим, что в некоторую 9-разрядную ячейку ОЗУ (8 информационных и один контрольный разряд) записывается код, представленный в строке 1 табл. 6.1. Контрольный разряд выбирается так, чтобы общее число единиц в 9-разрядном коде было нечетным. В данном случае число единиц в информационном слове равно пяти, следовательно, контрольный разряд $\overline{\text{mod2}}$ выбирается равным 0. В процессе дальнейшей работы процессор обратится к этой ячейке памяти с целью извлечения хранимого слова (иначе не было бы смысла записывать это слово в ОЗУ).

При чтении (см. строку 2) обнаружилось, что общее число единиц в 9-разрядном слове оказалось четным. Сравнив строки 1 и 2, можем заключить, что имеется ошибка в разряде 5 (вместо 0 считана 1), но система контроля и коррекции пока об этом «не знает» — пока она лишь зарегистрировала наличие ошибки. Далее включается механизм коррекции, который работает следующим образом. Считанное 9-разрядное слово данных, содержащее ошибку, инвертируется, например, процессором и вновь записывается в эту же ячейку памяти (см. строки 3 и 4). Сразу после этого выполняется чтение из этой же ячейки памяти. Поскольку в данном примере рассматривается постоянная ошибка, а именно ошибка типа «постоянная единица»

в разряде 5 данной ячейки памяти, повторное чтение даст результат, показанный в строке 5 табл. 6.1. При этом число единиц в считанном 9-разрядном коде вновь оказалось четным, но на этот раз достаточно проинвертировать считанный код, чтобы получить правильный результат, совпадающий с исходным кодом (см. строки 1 и 6).

Потери времени, связаиные с коррекцией, имеют место только при обращении к неисправным ячейкам ОЗУ. Избыточная аппаратура: 9-разрядная схема проверки четности и дополнительный разряд ОЗУ.

6.6. Устройство для обнаружения ошибок — двух последовательных записей в одну и ту же ячейку ОЗУ [58]

Разумно составленная программа не может предусматривать две последовательные записи в одну и ту же ячейку ОЗУ без промежуточного чтения, поскольку первая запись теряется и, следовательно, не было смысла ее выполнять. Схема регистрации таких ошибок приведена на рис. 6.7. Блок памяти ОЗУ1 предназначен для хранения

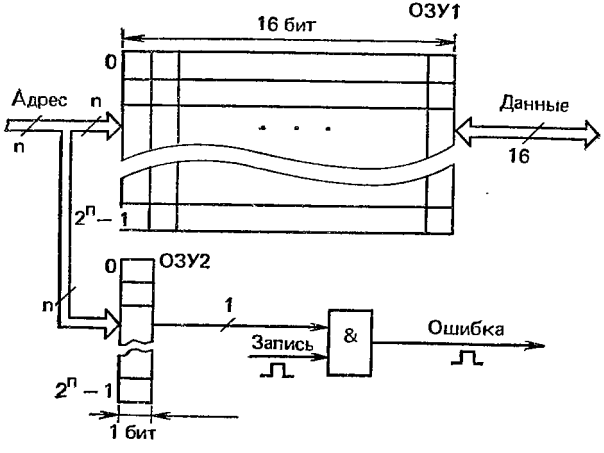


Рис. 6.7. Схема регистрации ошибок

информации. Блок ОЗУ2 имеет столько же ячеек, сколько и ОЗУ1, но каждая его ячейка содержит всего один разряд. Например, если ОЗУ1 имеет структуру 1М×16 бит, то ОЗУ2—1М×1 бит. Перед пуском программы во все ячейки ОЗУ2 записываются нули. При записи информации в ячейку ОЗУ1 производится чтение бита из соответствующей ячейки ОЗУ2, а затем (автоматически) — запись единицы в эту ячейку ОЗУ2. «Старое» значение бита поступает на элемент И. Если старое значение равно единице и выполняется режим записи,

то формируется сигнал ошибки. При чтении из ОЗУ1 соответствующий бит в ОЗУ2 устанавливается в 0.

Это же устройство может также реализовать проверку правильности выполнения программы или одного программного цикла по следующему критерию. Если можно заранее предсказать, какие ячейки памяти при выполнении программы будут обязательно прочитаны, то, обнаружив после выполнения программы среди них «непрочитанные» ячейки, можно утверждать, что программа выполнялась неправильно. Для регистрации таких ошибок все ячейки, которые должны быть прочитаны как минимум один раз, помечаются единицами в ОЗУ2. После окончания работы программы анализируется состояние ОЗУ2. Если в нем нет ни одной единицы, то программа прошла нормально (с точки зрения данного критерия).

6.7. Устройство для оперативного контроля магистрали микроЭВМ, использующее паузы в ее работе [59]

Если в отсутствие управляющего сигнала занятости BBSY (см. § 2.1) имеется достаточно времени для тестирования, в магистраль попеременно посылаются единичные и нулевые тестовые сигналы (рис. 6.8). Если в результате, например, пробоя входной цепи элемента-приемника (в любом устройстве) на одной из линий магист-

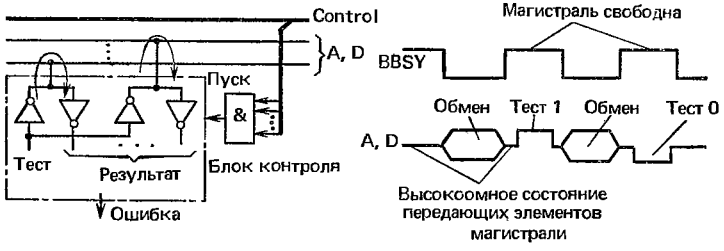


Рис. 6.8. Устройство для оперативного контроля магистрали

рали постоянно поддерживается высокий уровень напряжения, то не пройдет нулевой тестовый сигнал (Тест 0), а если в линии магистрали постоянно присутствует низкий уровень напряжения, то не пройдет единичный тестовый сигнал (Тест 1). При правильной работе результат будет совпадать с тестовым сигналом, при неправильной — будет сформирован сигнал ошибки (схема формирования сигнала ошибки не показана).

Запуск механизма тестирования производится только в том случае, когда ожидается достаточно длинная пауза, иначе тестовые сигналы могут помешать полезному обмену информацией по магистрали. В интерфейсе Общая шина условием, прогнозирующим появление достаточно длинной паузы, может являться одновременное отсутствие всех запросных сигналов BR_i, отсутствие сигналов SACK

и BBSY. При выполнении этого условия срабатывает элемент И и сигиал Пуск поступает на вход блока контроля.

Запуск механизма тестирования может производиться и «на общих основаниях», когда блок контроля периодически, например, с частотой 1 кГц «официально» захватывает магистраль и проводит сеанс тестирования. Можно конструктивно совместить блок контроля, например, с таймером, который периодически захватывает магистраль и посылает в процессор вектор прерывания. В этом случае, пока магистраль еще занята таймером, блок контроля проводит сеанс тестирования (но, разумеется, так, чтобы не мешать передаче вектора).

При реализации данного метода в коикретном устройстве целесообразно конструктивное разнесеиие передатчиков и приемников блока контроля к физическим концам магистрали. В этом случае будут также обнаруживаться ошибки типа «обрыв лииии магистрали».

6.8. Оперативный контроль магистрали микроЭВМ с возможностью адаптации к отказам информационных и адресных линий [60]

Рассматривается развитие метода контроля, описанного в предыдущем параграфе. Точно так же предполагается, что в те промежутки времени, когда магистраль свободна (высокий уровень напряжения на линии BBSY), срабатывает блок контроля, посылая поочередно в линии магистрали нулевые и единичные сигналы. Если в каком-либо байте обнаружена ошибка, то по линии контрольного разряда КР этого байта посылается отрицательный импульс (рис. 6.9). По приходу этого импульса срабатывают все блоки коррекции (на рисунке показан один такой блок), подключенные к линиям передачи данного байта.

Каждое устройство, подключенное к магистрали, снабжено блоками коррекции, по одному блоку на каждый принимаемый байт. Все блоки коррекции, приияв отрицательный импульс по линии КР в отсутствие занятости магистрали, записывают в регистр D7 код, содержащий единицу в том разряде, где зарегистрирована ошибка. Это происходит благодаря сравнению полученного кода с эталонным значением, снимаемым с выхода мажоритарного элемента D4 (единица на его выходе формируется тогда, когда по крайней мере на двух его входах присутствуют единицы). После загрузки регистров D7 все участники обмена «знают», где искать ошибку при нарушеиии условия общей нечетности числа едииниц в байте (с учетом КР).

Если при «полезном» обмене ошибка не зарегистрирована (передаваемый по отказавшей линии бит случайно совпал с ошибкой или линия самовосстановилась и сработала правильно), то группа логических элементов И (D8) закрыта сигналом с инвертора D5 и, следовательно, группа логических элементов Исключающее ИЛИ (D9) транслирует сигналы на выходы без инвертирования. Если 9-входовый элемент D1 (сумматор по модулю два) зарегистрировал при обмене четное число единиц, то это указывает на ошибку. Логические элементы D8 в этом случае передают код, содержащий единицу в «отказавшем» разряде, на элементы D9, в результате нужный

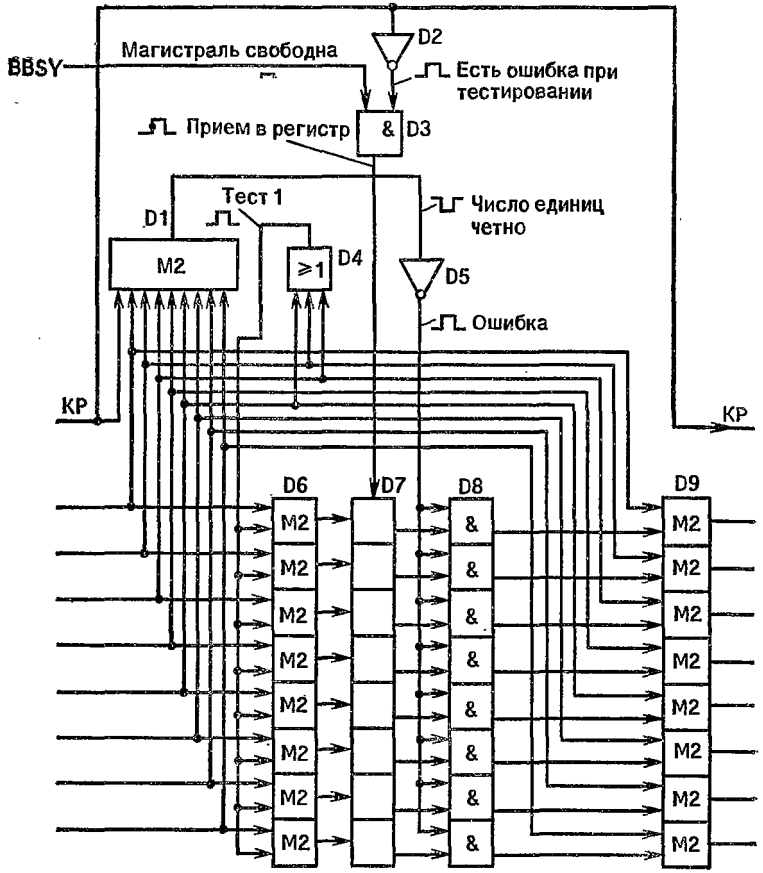


Рис. 6.9. Схема адаптации к ошибкам в магистрали

бит исправляется. Таким образом, микроЭВМ адаптируется к отказу линии магистрали или к неустойчивой работе линии. Отказ линии КР не приводит к искажению передаваемой информации.

6.9. Исправление постоянной ошибки в линии магистрали за счет «логической поддержки» ближайших соседних линий [61]

Упрощенная схема аппаратуры коррекции приведена на рис. 6.10. Предположим, что со стороны передатчика (левая часть схемы) в шину данных выдается код, представленный в строке 1 табл. 6.2,

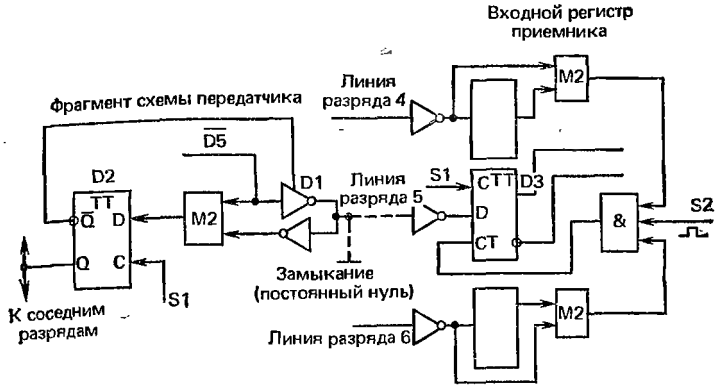


Рис. 6.10. Восстановление информации в магистрали при «поддержке соседей»

Предположим далее, что в линии D5 шины данных магистрали в результате, например, пробоя выходного транзистора одного из передатчиков постоянно поддерживается низкий уровень напряжения, что равносильно замыканию этой линии на «землю». В этом случае код в шине данных будет содержать одиночную ошибку, как показано в строке 2 табл. 6.2. Чтобы режим перегрузки передатчика D1 не длился слишком долго, производится его отключение. В триггер D2 по сигналу подтверждения истинности данных S1=1 принимается единица; это говорит о том, что попытка выдать в линию желаемую информацию не увенчалась успехом. Нулевой сигнал с плеча $\overline{Q}$ триггера D2 отключает перегруженный передатчик D1, предотвращая его выход из строя. Этот же сигнал S1 вызывает прием информации (с ошибкой) во входной регистр устройства-приемника. Устройство-приемник посылает в устройство-передатчик ответ-

Таблица 6.2

Номер кода	D7	D6	D5	D4	D3	D2	D1	D0	Примечание
1	1	0	1	1	0	0	1	1	Код для передачи абоненту
2	1	0	**0**	1	0	0	1	1	Код в магистрали (ошибка в разряде D5) и на входном регистре абонента
3	1	**1**	0	**0**	0	0	1	1	Проинвертированы соседние с ошибочным разряды в магистрали
4	1	0	1	1	0	0	1	1	Восстановленный код на входном регистре абонента

ный сигнал, подтверждающий факт приема информации. Фронт этого сигнала воздействует на устройство-передатчик, в результате это устройство инвертирует сигиалы в двух соседних с отказавшим разрядах, как показано в строке 3 табл. 6.2 (соответствующая аппаратура инвертирования сигналов на рис. 6.10 не показана). После окончания переходных процессов на «инвертированных» линиях передатчик снимает сигиал S1, в результате в приемиике формируется импульс S2. Так как к этому моменту информация, пришедшая с магистрали к двум соседним триггерам, не совпадает с ранее принятой, срабатывает элемент И и триггер D3 инвертирует свое состояние, получая импульс иа вход СТ. Таким образом, восстанавливается правильное значение ошибочного бита. Обмен завершается снятием ответного сигнала от абонента. Соседями разряда 7 считаются разряды 6 и 0; соседями разряда 0 — разряды 7 н 1, т. е. байт «замкнут в кольцо».

Рассмотренный метод позволяет исправлять групповые ошибки, если работоспособные соседи окружают каждый отказавший разряд. Поэтому целесообразно распределять выходные усилители передатчика так, чтобы каждая микросхема обслуживала иесмежиые линии магистрали. Тогда отказ всей микросхемы (группы усилителей) будет «раздроблен» на линиях магистрали и надежда на поддержку соседей оправдается. То же относится и к распределению линий между микросхемами магистральных приемников — отказ всей микросхемы не должен приводить к «сплошным» неисправным группам линий магистрали.

6.10. Коррекция одиночных отказов информационно-адресной шины путем инвертирования передаваемых кодов [62]

Если попытка передачи в магистраль иекоторого кода не увенчалась успехом, т. е. если передатчик «видит» в магистрали не тот бит, который он выдает (см. рис. 6.10, элемеит Исключающее ИЛИ (М2) на входе триггера D2), то он может адаптироваться к ошибке, повторив попытку выдачн, но выдав код в инверсном виде. В этом случае ошибка «прячется», совпадая по фазе с передаваемым битом: ошибка типа «постоянная единица» не мешает передать «настоящую» единицу, а ошибка типа «постояиный нуль» не проявляется при передаче нуля. Чтобы приемник зиал, в какой фазе идет передача, в магистраль вводится дополнительная линия, предпочтительно резервированная: если на этой линии, например, нуль, то передаваемый код представлен в прямом виде, а если на линии единица, то код передается в инверсиом виде. В последнем случае приемник инвертирует принятую информацию для получения правнльного результата.

6.11. Подключение абонентов к адресной шине с возможностью коррекции ошибок в ее линиях [63]

На рис. 6.11 в качестве примера показаны четыре абонеитских блока 1—4, входы которых 5—13 подключены к адресной шине 14, содержащей девять линий 15—23. Абонентский блок 1 (2, 3, 4) содержит три пары логически элементов: мажоритарный элемент 25 —

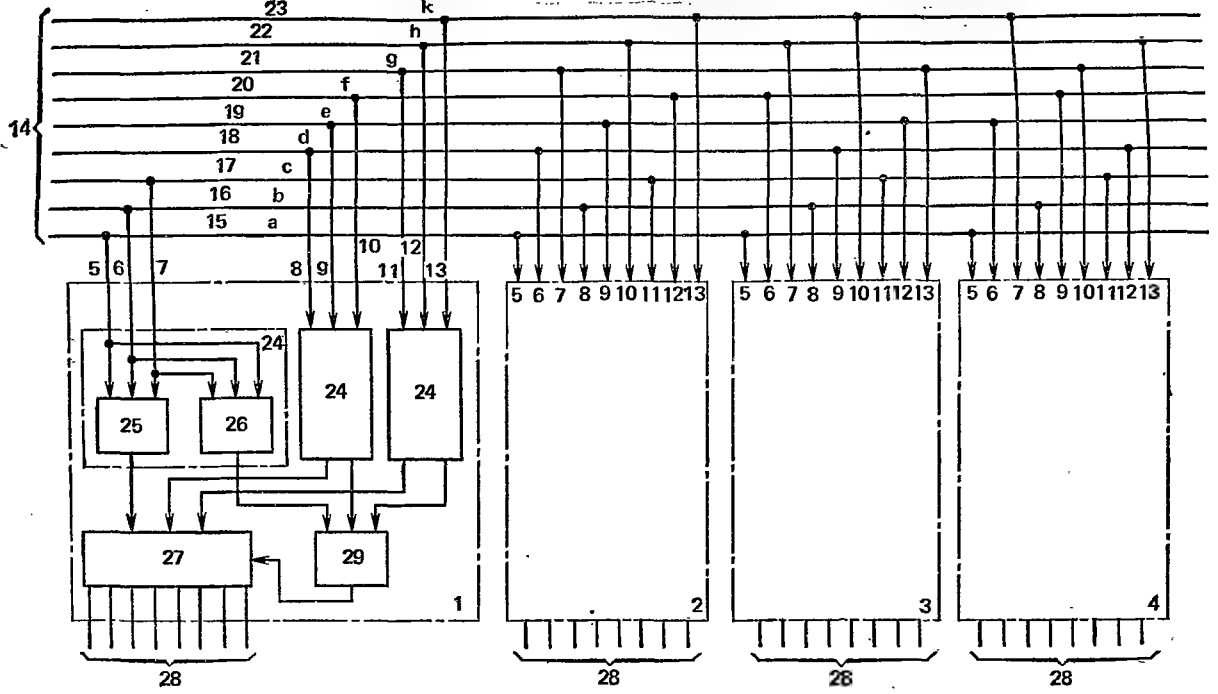


Рис. 6.11. Резервированнная система адресации

элемент сравнения 26. Устройство позволяет распознавать на 9-разрядной адресной шине 24 «индивидуальные» и 2 «общие» комбинации при наличии ошибки не более чем в одном разряде.

Для срабатывания дешифратора 27, например в блоке 1, необходимо, чтобы с выхода мажоритарного элемента 29 на управляющий вход этого дешифратора поступил сигнал лог. 1. При этом на одном из его выходов 28 в зависимости от кода на входах будет сформирована лог. 1. Элемент 29 срабатывает в тех случаях, когда на его входы поступили по крайней мере две единицы, т. е. в тех случаях, когда по крайней мере два элемента 26 из трех обнаружили на своих входах совпадение сигналов (комбинации 000 или 111).

Пусть, например, на входы 5—7 и 8—10 блока 1 поданы нули, а на входы 11—13 этого блока — единицы. Эти условия отражены в строке 1 табл. 6.3, из которой видно, что указанная в строке 1

Таблица 6.3

Номер строки	Входы блока 1			Входы блока 2			Входы блока 3			Входы блока 4		
	abc	def	ghk	adg	beh	cfk	afh	bdk	ceg	aek	bfg	cdh
1	000	000	111	001	001	001	001	001	001	001	001	001
2	000	111	000	010	010	010	010	010	010	010	010	010
3	000	111	111	011	011	011	011	011	011	011	011	011
4	111	000	000	100	100	100	100	100	100	100	100	100
5	111	000	111	101	101	101	101	101	101	101	101	101
6	111	111	000	110	110	110	110	110	110	110	110	110
7	001	001	001	000	000	111	010	001	100	001	010	100
8	010	010	010	000	111	000	001	100	010	010	100	001
9	011	011	011	000	111	111	011	101	110	011	110	101
10	100	100	100	111	000	000	100	010	001	100	001	010
11	101	101	101	111	000	111	110	011	101	101	011	110
12	110	110	110	111	111	000	101	110	011	110	101	011
13	001	010	100	001	010	100	000	000	111	010	001	100
14	010	100	001	010	100	001	000	111	000	001	100	010
15	011	110	101	011	110	101	000	111	111	011	101	110
16	100	001	010	100	001	010	111	000	000	100	010	001
17	101	011	110	101	011	110	111	000	111	110	011	101
18	110	101	011	110	101	011	111	111	000	101	110	011
19	001	100	010	010	001	100	001	010	100	000	000	111
20	010	001	100	001	100	010	010	100	001	000	111	000
21	011	101	110	011	101	110	011	110	101	000	111	111
22	100	010	001	100	010	001	100	001	010	111	000	000
23	101	110	011	110	011	101	101	011	110	111	000	111
24	110	011	101	101	110	011	110	101	011	111	111	000
25	000	000	000	000	000	000	000	000	000	000	000	000
26	111	111	111	111	111	111	111	111	111	111	111	111

комбинация сигналов является «благоприятной» для блока 1. Действительно, при отсутствии ошибок на выходах всех трех элементов сравнения 26 блока 1 будут сформированы лог. 1, мажоритарный элемент 29 формирует сигнал разрешения работы дешифратора, из одном из его выходов 28 (а именно, на выходе, соответствующем коду 001 на его входах) будет сформирована единица. Искажение любого бита на одиой из линий 15—23 не повлияет на результат, поскольку элементы 25 не пропустят ошибку на вход дешифратора 27, а элемент 29 сработает и в том случае, когда на его входах две, а не три единицы.

Из табл. 6.3 также следует, что указанная в строке 1 комбииация сигиалов является «неблагоприятной» для блоков 2—4 в том смысле, что ни в одном из них, даже при наличии одиночной ошибки, одновременно не сработают два элемента 26 и, следовательно, дешифраторы 27 в этих блоках не откроются. Таким образом, кодовая комбинация, указанная в строке 1 табл. 6.3, даже при наличии ошибки будет опознана единственным блоком — блоком 1, на одном из выходов которого сформируется единичный сигнал. Этот сигнал, как обычно, может использоваться для выбора информации из некоторого адресуемого элемента, для записи информации в этот адресуемый элемент или для других целей в зависимости от особенностей конкретной системы.

Продолжая рассмотрение табл. 6.3, можно убедиться в том, что кодовые комбинации, указанные в строках 2—6, распознаются блоком 1, комбинации, указанные в строках 7—12 (13—18, 19—24), распознаются блоком 2 (3, 4). Кодовые комбинации, указанные в строках 25—26, распознаются одиовремеино всеми абоиентскими блоками 1—4 и могут использоваться для посылки групповых команд Общий сброс, Конец связи и т. п.

6.12. Устройство для контроля управляющей вычислительной машины [64]

Во многих системах вызову той или иной подпрограммы управления объектом должен предшествовать сигнал прерывания по соответствующей управляющей линии магистрали. Если это так, то критерием неправильного обращения к i-й подпрограмме управления объектом является попадаиие адреса в зону, отведеиную для этой подпрограммы, при отсутствии предшествующего сигнала по i-й интерфейсной линии прерывания. Устройство, показанное на рис. 6.12, проверяет корректность обращения процессора к подпрограммам обслуживания объекта в соответствии с указаиным выше критерием.

Устройство приводится в исходное состояние при поступлении отрицательного импульса в линию общего сброса R магистрали M. При этом на выходах регистра прерываний устанавливаются лог. 1, на выходе триггера ошибки T формируется 0, что соответствует отсутствию ошибки.

При правильной работе системы обращению к подпрограмме управления объектом должен предшествовать сигнал прерывания иа соответствующей линии прерывания магистрали M. Предположим, что в результате появления отрицательного импульса на линии Z установлен в 1 триггер TrZ и, следовательно, на его нулевом выходе сформирован сигиал лог. 0. Этот сигнал закрывает соответствующий вход И элемента И—ИЛИ—НЕ, так что отрицательные импульсы

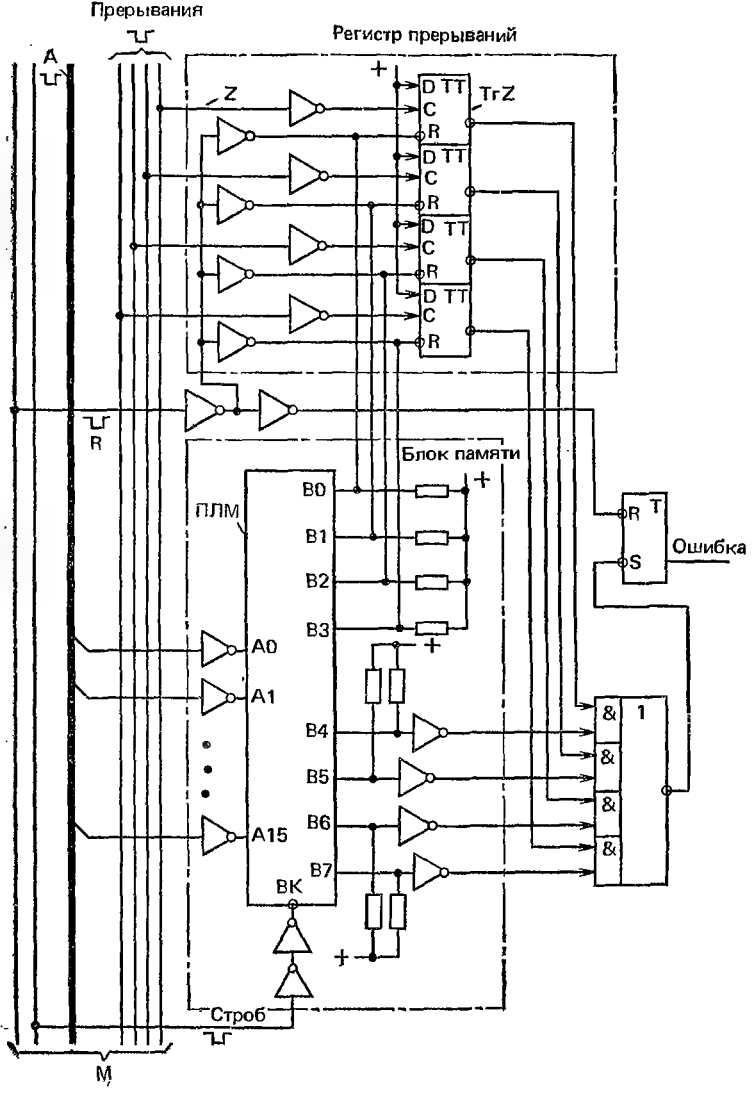


Рис. 6.12. Устройство дли защиты программ.

A0 — младший разряд адреса

с выхода B4 программируемой логической матрицы ПЛМ (см. § 1.4), пройдя через соответствующий инвертор, не приведут к установке триггера T по входу S.

Отрицательный импульс на выходе B4 формируется всякий раз, когда ПЛМ опознаёт принадлежность текущего адреса, поступающего из процессора по шине A магистрали M, к зоне адресов, отведенной для размещения подпрограммы управления объектом, связанной с установкой триггера TrZ. Кодировка ПЛМ такова, что выбор последней команды подпрограммы (команды выхода из прерывания) не сопровождается появлением отрицательного импульса на выходе B4; вместо него формируется импульс на выходе B0, который устанавливает в 0 триггер TrZ. Таким образом, при правильной работе системы триггер T остается в нулевом состоянии.

При попытке обращения к подпрограммам управления объектом без поступления сигнала прерывания регистр прерываний не препятствует прохождению сигналов с выходов B4—B7 ПЛМ на вход установки триггера T, поэтому на его выходе формируется сигнал ошибки, который может использоваться, например, для отключения напряжения питания от объекта или для принятия других экстренных мер с целью предотвращения аварии.

Рассмотрим пример задания кодировки ПЛМ по известным зонам расположения четырех подпрограмм П0—П3 управления объектом:

П0 зона 5560—5580;
П1 зона AA80—AAAA;
П2 зона AAAC—AAC0;
П3 зона AB00—AC00.

Условия сброса соответствующих триггеров регистра прерываний определяются обращением к последним ячейкам указанных зон:

$\overline{B0}=\overline{A15}\cdot A14\cdot\overline{A13}\cdot A12\cdot\overline{A11}\cdot A10\cdot\overline{A9}\cdot A8\cdot A7\cdot\overline{A6}\cdot\overline{A5}\cdot\overline{A4}\cdot\overline{A3}\cdot\overline{A2}\times$
$\times\overline{A1}\cdot\overline{A0}$ (знаки «·» и × эквивалентны);

$\overline{B1}=A15\cdot\overline{A14}\cdot A13\cdot\overline{A12}\cdot A11\cdot\overline{A10}\cdot A9\cdot\overline{A8}\cdot A7\cdot\overline{A6}\cdot A5\cdot\overline{A4}\cdot A3\cdot\overline{A2}\times$
$\times\overline{A1}\cdot\overline{A0}$;

$\overline{B2}=A15\cdot\overline{A14}\cdot A13\cdot\overline{A12}\cdot A11\cdot\overline{A10}\cdot A9\cdot\overline{A8}\cdot A7\cdot A6\cdot\overline{A5}\cdot\overline{A4}\cdot\overline{A3}\cdot\overline{A2}\times$
$\times\overline{A1}\cdot\overline{A0}$;

$\overline{B3}=A15\cdot\overline{A14}\cdot A13\cdot\overline{A12}\cdot A11\cdot A10\cdot\overline{A9}\cdot\overline{A8}\cdot\overline{A7}\cdot\overline{A6}\cdot\overline{A5}\cdot\overline{A4}\cdot\overline{A3}\cdot\overline{A2}\times$
$\times\overline{A1}\cdot\overline{A0}$.

Условия попадания в зоны П0—П3 (исключая последние ячейки):

$\overline{B4}=\overline{A15}\cdot A14\cdot\overline{A13}\cdot A12\cdot\overline{A11}\cdot A10\cdot\overline{A9}\cdot A8\cdot\overline{A7}\cdot A6\cdot A5$;

$\overline{B5}=A15\cdot\overline{A14}\cdot A13\cdot\overline{A12}\cdot A11\cdot\overline{A10}\cdot A9\cdot\overline{A8}\cdot A7\cdot\overline{A6}\cdot\overline{A5}\vee$
$\vee A15\cdot\overline{A14}\cdot A13\cdot\overline{A12}\cdot A11\cdot\overline{A10}\cdot A9\cdot\overline{A8}\cdot A7\cdot\overline{A6}\cdot A5\cdot\overline{A4}\cdot\overline{A3}\vee$
$\vee A15\cdot\overline{A14}\cdot A13\cdot\overline{A12}\cdot A11\cdot\overline{A10}\cdot A9\cdot\overline{A8}\cdot A7\cdot\overline{A6}\cdot A5\cdot\overline{A4}\cdot A3\times$
$\times\overline{A2}\cdot\overline{A1}$;

$\overline{B6}=A15\cdot\overline{A14}\cdot A13\cdot\overline{A12}\cdot A11\cdot\overline{A10}\cdot A9\cdot\overline{A8}\cdot A7\cdot\overline{A6}\cdot A5\cdot\overline{A4}\cdot A3\cdot A2\vee$
$\vee A15\cdot\overline{A14}\cdot A13\cdot\overline{A12}\cdot A11\cdot\overline{A10}\cdot A9\cdot\overline{A8}\cdot A7\cdot\overline{A6}\cdot A5\cdot A4$;

$\overline{B7}=A15\cdot\overline{A14}\cdot A13\cdot\overline{A12}\cdot A11\cdot\overline{A10}\cdot A9\cdot A8$.

В приведенном примере использовано 11 логических произведений, каждое из которых содержит не более 16 переменных, поэтому функции B0—B7 могут быть реализованы с помощью стандартной ПЛМ К556РТ1, см. § 1.4.

6.13. Передача управления при «зависаниях» путем доопределения магистральных сигналов [120]

Предполагается, что центральный процессор ЦП не имеет внутренних схем реакции на зависания (обращения по «несуществующим» адресам). Поэтому в микроЭВМ введены следующие элементы (рис. 6.13): группа резисторов R, таймер D и ключ SW.

При нормальной работе ключ (группа логических элементов) замкнут; сигналы a и b на выходах таймера отсутствуют, процессор, как обычно, ведет обмен с абонентами (памятью, контроллерами

Рис. 6.13. Схема регистрации зависаний

и т. д.). Резисторы R «не мешают» обмену, так как их сопротивление сравнительно велико (например, 10 кОм). Если адрес не опознан ни одним из абонентов в течение 10 мкс, то срабатывает таймер D и сигнал b размыкает ключ примерно на 100 мкс. В то же время в процессор посылается ответственный сигнал a, имитирующий ответ от абонента. Процессор продолжает работу, не зная о том, что ситуация усугубилась размыканием ключа. Поэтому последующие обращения вновь приведут к зависаниям даже при правильном указании адреса. Это делается для того, чтобы процессор, продвигаясь по микропрограмме, обязательно «добрался» до фазы обращения за очередной командой — как только это случится, он попадает в «ловушку».

Действительно, при разомкнутом ключе все абоненты находятся в высокоомном состоянии, а резисторы обеспечивают поддержание на шине данных кода команды безусловной передачи управления, например, в нулевую ячейку памяти. Обратившись в нулевую ячейку, процессор «видит» в ней ту же самую «распаянную» на шине данных команду, так как ключ по-прежнему разомкнут. При выполнении этой команды он вновь обращается в нулевую ячейку и т. д. При разомкнутом ключе циклы обращения к магистрали повторяются с периодом 10 мкс, при этом в каждом цикле сигнал a имитирует ответ от абонента. По истечении 100 мкс от первого зависания и при освобождении магистрали, если она в этот момент была занята, ключ замыкается, так что, обратившись в очередной раз в нулевую ячейку, процессор увидит в ней «настоящую» команду,

а именно первую команду программы реакции на зависание. В этой программе предусматривается, например, выдача сообщения оператору и вызов диагностических тестов.

6.14. Контроль сохранности данных в статическом ОЗУ с резервным источником питания

Процесс сохранения данных в статическом КМОП ОЗУ с резервным источником питания при отключении напряжения первичной питающей сети был рассмотрен в гл. 2 (рис. 2.7, 2.8). Благодаря сохранению данных, их можно использовать при дальнейшей работе микроЭВМ после включения напряжения первичной питающей сети.

При включении напряжения питания микроЭВМ процессор автоматически переходит к выполнению стандартной процедуры начального пуска. Обычно эта процедура определяется программой, хранимой в ПЗУ. В ходе выполнения этой программы процессор проверяет, была ли у микроЭВМ «предыстория» или машина включена «впервые». Если процессор определил, что информация в статическом ОЗУ отражает работу микроЭВМ до выключения напряжения питания, то информация считается достоверной и используется при дальнейшей работе. Если у процессора есть основания сомневаться в достоверности хранимой в статическом ОЗУ информации, то он не использует ее и переходит к выполнению ветви программы начального пуска, соответствующей «первому» включению микроЭВМ (не учитывающему предысторию).

Прежде чем использовать сохраненную в статическом ОЗУ информацию, процессор должен оценить ее достоверность. Оценка может быть приближенной или точной, в зависимости от требований, предъявляемых к надежности функционирования микроЭВМ в конкретной системе. Повышение точности оценки связано с использованием дополнительных аппаратных или (и) программных средств. Рассмотрим сначала методы косвенного контроля достоверности информации, сохраненной в статическом ОЗУ в период отсутствия напряжения первичной питающей сети.

1. Контроль состояния статического ОЗУ по содержимому отдельных ячеек [121, 122]. В системах, к которым не предъявляются повышенные требования к надежности функционирования, можно оценивать достоверность сохраненной информации по содержимому нескольких ячеек статического ОЗУ. Например, в предыстории, задолго до выключения напряжения питания, процессор заранее записал в специально отведенные для этого 64 ячейки статического ОЗУ некоторые характерные коды (но не «все нули» или «все единицы»). Копии этих кодов (или алгоритм их формирования) хранятся в ПЗУ. В некоторый момент напряжение питания выключается. Благодаря подключению статического ОЗУ к резервной батарее, информация (в том числе и 64 контрольных кода) сохраняется.

При последующем включении напряжения питания процессор, выполняя хранимую в ПЗУ программу, считывает контрольные коды из статического ОЗУ и сравнивает их с эталонными кодами, хранящимися в ПЗУ (или полученными путем вычислений по программе, хранимой в ПЗУ). Если хотя бы один из указанных кодов не совпал с эталонным, то вся сохраненная в статическом ОЗУ информация считается недостоверной. Если все 64 кода совпали с эталонными,

то предполагается, что и в остальных ячейках ОЗУ хранится правильная информация. Такое предположение, конечно, рискованно — по содержимому отдельных ячеек нельзя с уверенностью судить о правильности всей остальной информации. Напомним, однако, что сбой микроЭВМ, как предполагается, не приводит к аварии или иным нежелательным ситуациям (если микроЭВМ используется, например, в игровом автомате).

2. Контроль состояния статического ОЗУ по смысловому содержанию имеющейся в нем информации [123]. В качестве примера рассмотрим систему, в которой микроЭВМ встроена в портативный медицинский прибор для обследования пациента. Перед началом обследования врач с помощью клавиатуры вводит в микроЭВМ анкетные данные пациента и другие сведения, которые запоминаются в статическом ОЗУ. Далее начинается процесс обследования, при котором информация считывается с датчиков, обрабатывается и в текстовой форме накапливается в статическом ОЗУ. Обследование, возможно, ведется с перерывами, при которых основной источник питания микроЭВМ выключается. (Напомним, что сохранность информации в статическом ОЗУ обеспечивается подключением резервной батареи.)

При включении напряжения питания процессор микроЭВМ анализирует область статического ОЗУ, отведенную для накопления текстовой информации о результатах обследования и других сведений. Текущая граница области помечена в ОЗУ определенным символом или сочетанием символов. Процессор последовательно просматривает все коды, имеющиеся в указанной области, и определяет их принадлежность определенному алфавиту (системе кодирования ASCII). Если все коды соответствуют символам, принятым в системе ASCII, то информация считается достоверной и обследование продолжается; результаты накапливаются в статическом ОЗУ и после обследования могут быть, например, выданы на печать или (и) по телефонному каналу связи переданы в центральный банк данных. При преднамеренном стирании информации статическое ОЗУ заполняется «запрещенными» (не используемыми в таблице кодирования ASCII) кодами.

Если при последовательном просмотре кодов (при включении напряжения питания) встретился хотя бы один «запрещенный» код, то это означает, что исходная информация еще не введена с клавиатуры или произошла потеря полезной информации при ее хранении из-за разряда или отключения резервной батареи, сбоя или отказа микроЭВМ и т. п.

Достоверность контроля может быть повышена, например, путем дополнительного выявления орфографических и синтаксических ошибок в хранимой текстовой информации. Так, используемые «служебные» слова должны совпадать по написанию с хранимыми в ПЗУ эталонами, между словами должны быть промежутки и т. п.

3. Контроль степени разряда резервной батареи как показателя недостоверности хранимой в статическом ОЗУ информации. Если микросхемы статической памяти, согласно техническим условиям, работают в диапазоне от 3 до 15 В, то при разряде резервной батареи до уровня, при котором напряжение питания ОЗУ достигает 3 В и продолжает снижаться, нельзя гарантировать сохранность информации.

На рис. 6.14 показана схема микроЭВМ, в которой при вклю-

чении напряжения питания проверяется уровень разряда резервной батареи.

В отсутствие наприжения 220 В питание статического ОЗУ осуществляется от резервной батареи через диод VD2. Диод VD1 и транзистор VT закрыты, вход «+» компаратора представляет собой высокое сопротивление. При включении напряжения 220 В блок питания формирует стабилизированные напряжения питания, диод VD1 открывается, диод VD2 закрывается, транзистор VT остается выключенным, так как на выходе E выходного порта микроЭВМ сформировано напряжение низкого уровня (E=L). Ток от батареи практически не потребляется.

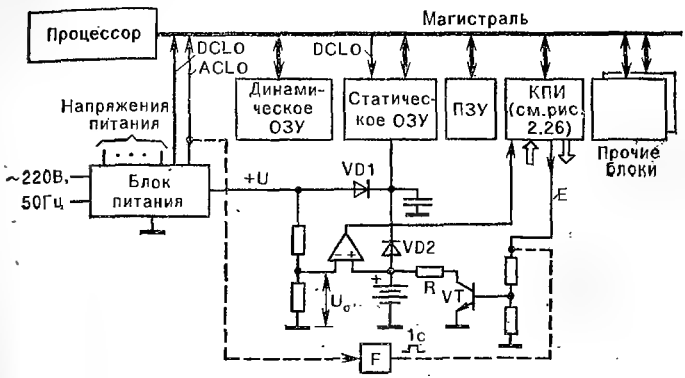


Рис. 6.14. Подключение аппаратуры контроля напряжения батареи к микроЭВМ

После установления номинальных значений питающих напряжений блок питания формирует напряжения высокого уровня на линиях ACLO и DCLO (см. рис. 2.7) управляющей шины магистрали. Процессор переходит к выполнению программы начального пуска, хранимой в ПЗУ. Эта программа, в частности, предусматривает выдачу сигнала E=H в линию выходного порта, выделенную для управления транзистором VT. Этот транзистор насыщается и подключает к батарее нагрузочный резистор R для создания тока, равного (или несколько большего) току, потребляемому статическим ОЗУ от батареи при выключенном блоке питания. Компаратор сравнивает напряжение на нагруженной батарее с опорным напряжением U_O, равным 4 В. (Напряжение на микросхемах статического ОЗУ в режиме питания от батареи не должно быть ниже 3 В; с учетом прямого падения напряжения на диоде VD2, не превышающего 1 В, напряжение на батарее не должно быть ниже 4 В.)

Если батарея обеспечивает нужное напряжение, то на выходе компаратора формируется сигнал лог. 1, в противном случае — сигнал лог. 0. В программе начального пуска предусмотрен опрос состояния компаратора с использованием одного из разрядов входного порта микроЭВМ. В зависимости от результата опроса процессор считает данные либо достоверными, либо нет. После опроса компа-

ратора в выходной порт выдается сигнал E=L, транзистор VT выключается и поддерживается в выключенном состоянии на протяжении всей дальнейшей работы микроЭВМ, так что ток от батареи практически не потребляется.

Чтобы не занимать линию выходного порта, можно использовать одновибратор F, который на время, равное, например, 1 с после включения блока питания, подключает нагрузочный резистор R к батарее, а в дальнейшем отключает его. За время, пока батарея находится под нагрузкой, программа начального пуска производит опрос выхода компаратора. Отметим, что после подключения нагрузочного резистора ток от батареи принимает установившееся значение не сразу и опрос выхода компаратора следует проводить с задержкой, зависящей от типа используемой батареи.

В [153] предлагается оценивать степень разряда батареи следующим образом. Перед выключением основного напряжения питания в статическом ОЗУ создаются два одинаковых массива сохраняемой информации с одинаковыми контрольными суммами. После выключения и последующего включения основного напряжения питания процессор выполняет программу пуска и проверяет сохранность обоих массивов, сверяя вычисленные контрольные суммы с хранимыми. Если оба массива не содержат ошибок, то батарея считается полностью работоспособной. Если в одном из двух массивов обнаружено несовпадение вычисленной и хранимой контрольных сумм, то этот массив уничтожается и на его место помещается копия «исправного» массива. Оператору выдается предупреждение о том, что батарею следует заменить, хотя микроЭВМ еще работоспособна. Если оба массива содержат ошибки, то информация считается утраченной, оператору выдается сообщение о необходимости замены батареи для обеспечения работоспособности микроЭВМ в режиме сохранения информации.

Рассмотренные методы косвенного контроля могут использоваться совместно в любых сочетаниях. Перейдем к методам непосредственного контроля.

1. Метод контрольного суммирования. Этот метод широко используется в вычислительной технике. Коды, образующие контролируемый массив, последовательно арифметически суммируются, причем перенос из старшего разряда не теряется, а складывается с младшим разрядом полученной суммы. Окончательная сумма запоминается. (Возможны и другие способы формирования контрольной суммы.)

В дальнейшем, при проверке сохранности массива кодов суммирование повторяют и сравнивают новую контрольную сумму со старой. Совпадение контрольных сумм позволяет с высокой достоверностью считать, что все коды сохранены без искажений.

Было бы логично перед выключением напряжения питания микроЭВМ запомнить в статическом ОЗУ информацию, которая может пригодиться в будущем и просуммировать (с циклическим переносом) все коды, хранящиеся в этом ОЗУ, за исключением кода из последней ячейки, отведенной для записи в нее окончательной контрольной суммы. На подсчет контрольной суммы может потребоваться время порядка 100 мс. После подсчета можно выключать напряжение питания микроЭВМ (статическое ОЗУ получает энергию от резервной батареи). При последующем включении напряжения питания микроЭВМ процессор подсчитывает новую контрольную сумму и сверяет ее со старой. В зависимости от результата сравнения

он переходит на одну из двух ветвей программы начального пуска.

Таким образом, для использования метода контрольного суммирования необходимо заранее предупреждать процессор микроЭВМ о намерении оператора выключить напряжение питания, так как при выключении без предупреждения у процессора остается слишком мало времени (5 мс вместо требуемых 100 мс, см. рис. 2.7) для подсчета контрольной суммы всех кодов, размещенных в статическом ОЗУ. С другой стороны, нет оснований требовать от человека-оператора безусловного выполнения «предупреждающих» действий (ввода с клавиатуры определенной команды, нажатия специальной кнопки или проч.) перед выключением напряжения питания микроЭВМ — это усложняет работу с машиной и не гарантирует сохранность информации из-за возможных ошибочных действий оператора.

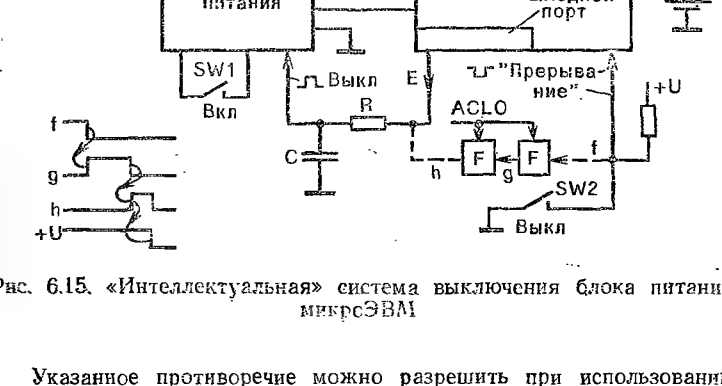


Рис. 6.15. «Интеллектуальная» система выключения блока питания микроЭВМ

Указанное противоречие можно разрешить при использовании схемы, приведенной на рис. 6.15. Оператор может беспрепятственно включить напряжение питания микроЭВМ кратковременным замыканием ключа SW1, однако выключить напряжение без разрешения на это со стороны микроЭВМ нельзя. Для выключения напряжения питания оператор кратковременно замыкает ключ SW2. При этом в микроЭВМ посылается сигнал прерывания, приоритет которого достаточно высок для безусловного восприятия процессором в течение времени, не превышающего нескольких секунд.

Восприняв сигнал прерывания, процессор завершает работу со статическим ОЗУ, подсчитывает контрольную сумму, записывает ее в определенную ячейку этого ОЗУ и выдает в выходной порт сигнал E=H (напряжение высокого уровня). Этот сигнал проходит через RC-цепь и воздействует на блок питания, выключая его. При выключении формируются сигналы ACLO, DCLO (см. рис. 2.7). Статическое ОЗУ закрывается по входам сигналом DCLO, чтобы при выходе питающих напряжений за допустимые границы в период их «угасания» не было испорчено содержимое одной или нескольких ячеек. К моменту получения сигнала DCLO процессор должен находиться в пассивном состоянии, чтобы не был «сорван» обмен по ма-

гистрали, что могло бы привести к искажению информации в статическом ОЗУ из-за непредсказуемости сочетаний сигналов при таком срыве.

При последующем включении напряжения питания процессор переходит к выполнению программы начального пуска, в которой, в частности, предусмотрена выдача в выходной порт сигнала E=L. Возможный кратковременный «всплеск» напряжения (E=H) при включении блока питания компенсируется интегрирующей RC-цепью.

Чтобы не занимать разряд выходного порта, можно использовать два последовательно включенных одновибратора F. При замыкании ключа SW2 на выходе цепи из одновибраторов с задержкой, равной, например, 3 с, формируется импульс с длительностью, достаточной для прохождения через RC-цепь и восприятия блоком питания. Предполагается, что за указанное время (3 с) процессор успевает отреагировать на сигнал прерывания и выполнить все необходимые действия по обслуживанию статического ОЗУ. При включении напряжения питания одновибраторы устанавливаются в пассивные состояния сигналом ACLO, поступающим из блока питания, чтобы не произошло ложного выключения.

Недостаток рассмотренного метода состоит в том, что при исчезновении напряжения питающей сети (в помещении) сохранность информации не гарантируется, контрольная сумма не вычисляется. То же относится и к «неправильному» выключению микроЭВМ, например, когда ключ SW2 не используется, а вилка шнура питания просто вынимается из розетки. Поэтому использовать данный метод можно в системах, допускающих потерю информации без серьезных последствий.

Далее рассмотрены методы контроля, лишенные указанного недостатка.

2. Метод постраничного контрольного суммирования. Как было показано, при выключении напряжения питания микроЭВМ у процессора остается слишком мало времени, чтобы успеть просуммировать все коды, хранящиеся в статическом ОЗУ. Поэтому, чтобы при выключении напряжения питания суммировать не все коды, предлагается разделить (условно) все статическое ОЗУ на страницы, например, по 64 ячейки, и на каждой странице выделить одну ячейку для хранения в ней соответствующей контрольной суммы (рис. 6.16).

При нормальной работе микроЭВМ все страницы, за исключением, возможно, одной, которая находится в процессе обновления, содержат правильные суммы S(N). Если выключение напряжения питания микроЭВМ застало процессор при работе по обновлению содержимого некоторой страницы, то эта «нетрудоемкая» работа успевает завершиться за оставшиеся 5 мс (см. рис. 2.7), так что к моменту перехода мнкроЭВМ на питание от резервной батареи все страницы содержат информацию с правильными контрольными суммами. В дальнейшем, при включении напряжения питания микроЭВМ процессор проверяет правильность информации на каждой странице, вычисляя новые контрольные суммы и сверяя их со старыми.

Для правильной работы микроЭВМ с предлагаемой страничной организацией статического ОЗУ необходимо выполнение ряда условий.

Операционная система должна рассматривать статическое ОЗУ как внешнее устройство (например, с магнитным носителем данных), в котором хранится ряд страниц. Произвольный доступ к ячейкам

статического ОЗУ (по крайней мере, по записи) должен быть запрещен. Для обработки информации, размещенной на некоторой странице, эта страница как единое целое (вместе с контрольной суммой) переписывается из статического ОЗУ в динамическое, где, таким образом, образуется рабочая копия. Обработка информации, т. е. преобразование рабочей копии из «старой» в «новую», производится с использованием динамического ОЗУ. При этом страница-оригинал не подвергается никаким изменениям. Для новой копии

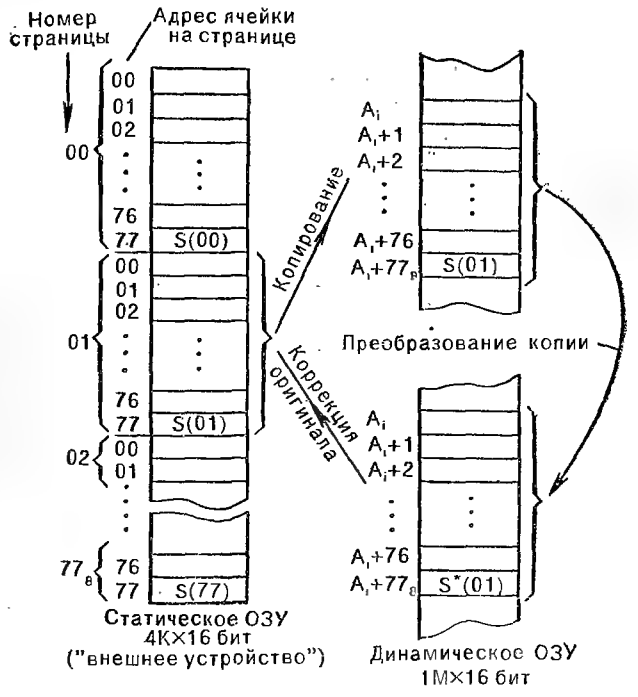


Рис. 6.16. Содержимое статического и динамического ОЗУ.

Использование статического ОЗУ как внешнего устройства позволяет поддерживать правильные значения всех контрольных сумм S(N) (N — номер страницы), за исключением, возможно, одной, к моменту выключения напряжения первичной питающей сети

вычисляется контрольная сумма S* (01), после чего проводится коррекция оригинала, т. е. перенос информации из динамического ОЗУ в статическое. В процессе переноса старая контрольная сумма теряет смысл, так как массив информации меняется. Однако переходный процесс завершается относительно быстро, что и требуется.

При «первом» включении напряжения питания микроЭВМ в статическом ОЗУ находится случайная информация с неправильными

контрольными суммами на всех страницах. Процессор в этой ситуации проводит начальную загрузку всех страниц статического ОЗУ эталонными (но не нулевыми) массивами кодов с правильными контрольными суммами.

3. Методы программного и схемного дублирования информации. Программы, использующие статическое ОЗУ для накопления данных, можно сотвить так, что слово при записи будет дублироваться, т. е. попадет в две ячейки, например, расположенные рядом. Это увеличивает время выполнения таких программ, но позволяет после выключения и включения напряжения питания микроЭВМ проверить правильность хранимой в статическом ОЗУ информации путем попарного сравнения содержимого указанных ячеек. При «первом» включении напряжения питания микроЭВМ процессор заполняет статическое ОЗУ эталонными дублированными кодами.

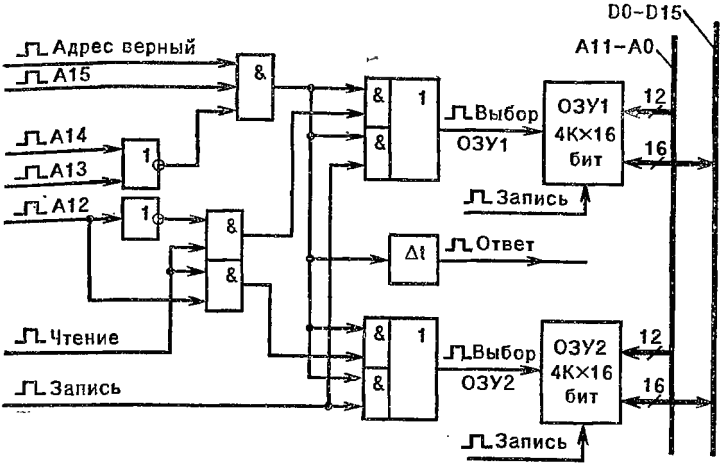
Возможны также различные схемные решения дублированной памяти. На рис. 6.17 показан вариант построения дублированной статической памяти и ее размещение в адресном пространстве микроЭВМ. При записи информации оба блока статической памяти ОЗУ1 и ОЗУ2 включены параллельно и занимают в адресном пространстве область 8000—8FFF и 9000—9FFF (на рис. 6.17, *б* показана первая из этих областей). При считывании блоку ОЗУ1 соответствует область 8000—8FFF, а блоку ОЗУ2 — область 9000—9FFF. При записи кода по адресу, лежащему в диапазоне 8000—8FFF, этот код одновременно попадает в одноименные ячейки ОЗУ1 и ОЗУ2. Считывание проводится раздельно из ОЗУ1 и ОЗУ2 по разным адресам.

При «первом» включении напряжения питания микроЭВМ процессор заполняет все ячейки с адресами 8000—8FFF произвольной (но не нулевой или единичной) информацией. Эта информация автоматически записывается параллельно в блоки ОЗУ1 и ОЗУ2, так как разряд адреса A12 не участвует в дешифрации (A15 — старший разряд полного 16-разрядного адреса). Далее процессор, не «зная» о существовании ОЗУ2, работает только с ОЗУ1 без каких-либо ограничений. Дублирующая память ОЗУ2 при этом всегда содержит ту же самую информацию, что и ОЗУ1, хотя она «скрыта» от процессора.

При выключении напряжения питания микроЭВМ не требуется вычислять контрольные суммы или выполнять прочие подобные операции, так как в ОЗУ2 к этому моменту имеется точная копия информации, хранимой в ОЗУ1, благодаря синхронной записи в оба блока. При последующем включении напряжения питания микроЭВМ процессор проверяет совпадение информации, сохраненной в блоках ОЗУ1 и ОЗУ2. Если ошибок во время хранения не возникло, то информация в ячейках 8000, 8001, 8002 и т. д. должна соответственно совпадать с информацией, хранимой в ячейках 9000, 9001, 9002 и т. д.

Вместо дублирования можно использовать «троирование», программное или аппаратное, по аналогии с приведенными здесь решениями. При этом появляется возможность коррекции ошибок.

4. Использование контролирующих кодов. При введении в статическое ОЗУ дополнительных контрольных разрядов можно обеспечить проверку правильности информации, сохраненной в отсутствие напряжения питания микроЭВМ. Каждому хранимому слову или байту ставится в соответствие один или несколько контрольных разрядов. Можно осуществлять контроль по четности (нечетности), по модулю 3, по модулю 7, по Хэммингу и т. д. Дополнительные аппаратные затраты при использовании контролирующих кодов мо-



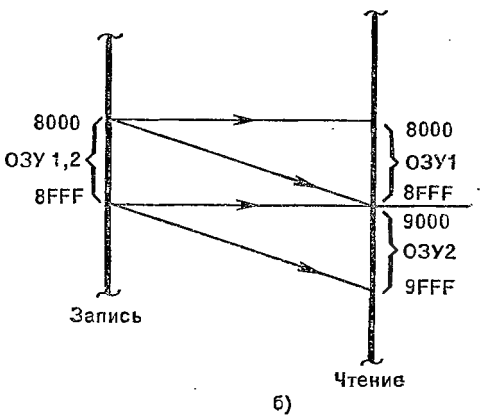


Рис. 6.17. Схема дублированного статического ОЗУ *(а)* с общей записью и раздельным считыванием; распределение адресного пространства *(б)* при записи и считывании информации

гут оказаться бо́льшими, чем при использовании иных методов; следует учесть также задержки, вносимые схемами формирования контрольных разрядов и схемами контроля.

5. Метод аппаратной коррекции контрольной суммы массива при его обновлении. При рассмотрении метода контрольного суммирования (см. п. 1) отмечалось, что проблема заключается в том, что при выключении напряжения питания микроЭВМ у процессо а остается

слишком мало времени (менее 5 мс), чтобы вычислить контрольную сумму всех кодов, хранимых в статическом ОЗУ. Поэтому предлагается ввести дополнительную аппаратуру, которая автоматически корректирует первоначально подсчитанное процессором значение контрольной суммы по мере записи новых кодов в статическое ОЗУ [75]. Таким образом, к моменту выключения напряжения питания правильное значение контрольной суммы известно и его достаточно лишь запомнить в одной из ячеек статического ОЗУ.

На рис. 6.18 показана схема устройства для поддержания правильного значения контрольной суммы в регистре RG1 при работе со статическим ОЗУ. Для простоты предполагается, что контрольная

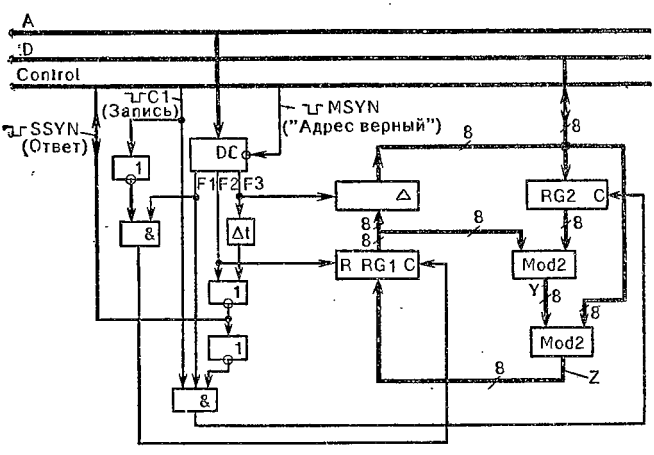


Рис. 6.18. Устройство для поддержания правильной контрольной суммы изменяемого массива информации

сумма вычисляется поразрядным суммированием по модулю два всех кодов контролируемого массива.

Дешифратор DC анализирует адресные коды, передаваемые процессором по шине A магистрали типа Общая шина (см. гл. 2), и вырабатывает сигналы F1, F2 и F3 в следующих случаях. Сигнал F1 (лог. 1) вырабатывается при обращении процессора к любой ячейке статического ОЗУ (процессор и статическое ОЗУ на рис. 6.18 не показаны с целью его упрощения). Сигнал F2 вырабатывается в ответ на обращение процессора по определенному адресу, не занятому другими адресуемыми элементами микроЭВМ. Сигнал F2 воздействует на вход R установки нуля регистра RG1, проходит через элемент ИЛИ—НЕ и поступает в управляющую шину Control магистрали в качестве ответного сигнала SSYN. Сигнал F3 вырабатывается дешифратором в ответ на обращение процессора к устройству с целью считывания содержимого регистра RG1. Сигнал F3 открывает усилители, содержимое регистра RG1 выдается в шину данных D магистрали, при этом сигнал с выхода элемента задержки

проходит через элемент ИЛИ—НЕ и подтверждает истинность выданных данных.

Для упрощения аппаратуры в схеме не предусмотрена начальная загрузка правильного значения вычисленной процессором контрольной суммы непосредственно с шины данных в регистр RG1. Загрузка выполняется в четыре этапа.

1) Процессор записывает в ячейку статического ОЗУ, выделенную для хранения контрольной суммы, нулевой код. При этом в регистр RG1 по фронту сигнала на его входе синхронизации C принимается некоторый код.

2) Процессор считывает из той же ячейки ОЗУ записанный ранее нулевой код. При выполнении считывания C1=H (напряжение высокого уровня), F1=H, SSYN=L (есть ответ от статического ОЗУ). Поэтому срабатывает трехвходовой элемент И и по фронту сигнала на его выходе в регистр RG2 с шины D принимается считанный из статического ОЗУ нулевой код.

3) Процессор выполняет операцию чтения или записи по адресу, в ответ на который дешифратор формирует сигнал F2; регистр RG1 устанавливается в 0.

4) Процессор выполняет команду записи вычисленной ранее контрольной суммы в ячейку статического ОЗУ, выделенную для ее хранения. На выходе двухвходового элемента И формируется сигнал приема информации в регистр RG1. В этот регистр принимается результат поразрядного суммирования по модулю два трех кодов: (а) старого содержимого регистра RG1 (нулевой код), (б) содержимого регистра RG2 (нулевой код) и (в) кода контрольной суммы, выданной процессором в шину данных для записи в ячейку статического ОЗУ. В результате в регистр RG1 заносится правильное начальное значение контрольной суммы.

Далее контрольная сумма корректируется по мере изменения содержимого контролируемого массива. Текущая коррекция значения контрольной суммы в регистре RG1 производится следующим образом. Программа составляется так, что каждой записи в ячейку контролируемого массива предшествует чтение из этой ячейки. (Это условие может быть перенесено на уровень микропрограмм процессора, в которых при обращении к статическому ОЗУ операция Запись заменяется операцией Чтение—модификация—запись (см. гл. 2); при такой замене указанное требование к программе снимается.)

При чтении информации из статического ОЗУ копия считанного байта запоминается в регистре RG2. На выходах первого каскада поразрядных сумматоров по модулю два формируется промежуточное (не старое, но еще и не новое) значение Y контрольной суммы, учитывающее тот факт, что только что считанный из ОЗУ код, возможно, будет уничтожен последующей записью нового кода. Иными словами, промежуточное значение контрольной суммы соответствует массиву, в котором считанный код заменен нулевым, если он таковым не является. Далее, при записи нового кода в ту же самую ячейку статического ОЗУ промежуточная контрольная сумма Y поразрядно складывается по модулю два с кодом, установленным процессором на шине D данных, в результате новая контрольная сумма Z принимается в регистр RG1.

Рассмотрим пример, поясняющий процесс коррекции контрольной суммы. Предположим, что статическое ОЗУ содержит 2К 8-разрядных ячеек. Для хранения контрольной суммы выделена последняя ячейка. Однако код в этой ячейке не соответствует правильному

значению контрольной суммы, так как массив информации постоянно обновляется за счёт записи новых кодов на место старых. Пусть в ОЗУ хранится следующий массив кодов (табл. 6.4):

Таблица 6.4

Номер ячейки	Содержимое	
0	0 1 0 1 1 1 0 0	
1	1 1 1 0 0 1 1 0	
2	0 1 1 0 1 1 0 1	
3	0 0 0 0 1 1 1 1	
4	0 0 0 0 0 0 0 0	Нулевые коды
5	0 0 0 0 0 0 0 0	
.	. . . .	
2046	0 0 0 0 0 0 0 0	
2047	××××××××	— В ячейке, выделенной для контрольной суммы, хранится произвольный код
Содержимое RG1:	1 1 0 1 1 0 0 0	— Правильная контрольная сумма

Предположим, что в ячейку с номером 3 вместо кода 00001111 процессор намерен записать код 11101100. Текущее значение контрольной суммы хранится в регистре RG1. При считывании старого кода из ячейки с номером 3 этот же код, как было показано, запоминается в регистре RG2. При этом формируется промежуточное значение Y контрольной суммы:

⊕11011000 — Старое значение контрольной суммы в регистре RG1
00001111 — Код, считанный в регистр RG2 из ячейки с номером 3
11010111 — Промежуточное значение контрольной суммы (Y)

При записи нового кода в ячейку с номером 3 промежуточная контрольная сумма Y поразрядно складывается по модулю два с кодом, который установлен процессором на шине D:

⊕11010111 — Промежуточное значение контрольной суммы (Y)
11101100 — Новый код, записываемый процессором в ячейку с номером 3
00111011 — Новое значение контрольной суммы в регистре RG1 (Z)

Таким образом, в регистре RG1 поддерживается правильное значение контрольной суммы, которое при выключении напряжения питания микроЭВМ программно переписывается в ячейку статического ОЗУ с номером 2047. После этого работа со статическим ОЗУ прекращается, так что в дальнейшем, к моменту включения напряжения питания микроЭВМ, контрольная сумма в ячейке 2047 соответствует массиву кодов, размещенному в ячейках 0—2046, что и проверяется процессором при выполнении программы начального пуска.

Вместо схем суммирования по модулю два могут быть использованы другие схемы, обеспечивающие более высокую достоверность контроля.

6.15. Устройство для коррекции ошибок при передаче или хранении информации [124]

Чтобы исправить ошибки, возникающие при передаче или хранении информации, необходимо ее избыточное представление с использованием корректирующих кодов. Идеальный корректирующий код должен обеспечивать простую и быструю дешифрацию при небольшом числе дополнительных (контрольных) разрядов. Однако эти требования во многом противоречивы. Предлагаемый в [124]

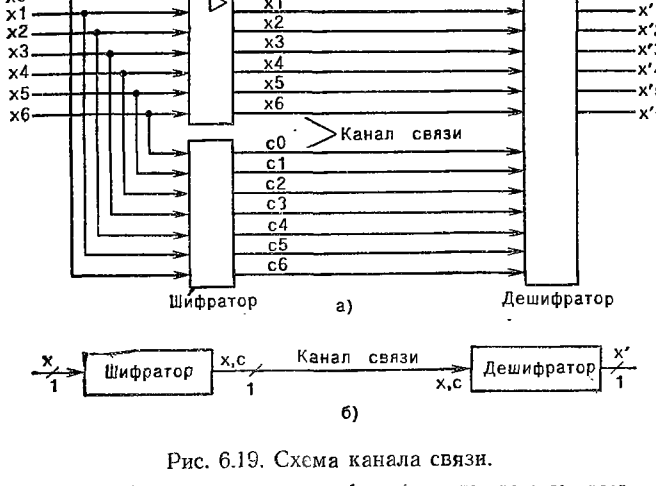


Рис. 6.19. Схема канала связи.

При передаче информации передатчик формирует контрольные разряды параллельным (*а*) или последовательным (*б*) кодом; приемник восстанавливает информационные биты, если в канале связи возникли ошибки допустимой кратности

метод построения «обобщенных ортогональных кодов» (GOC) обеспечивает простую и быструю дешифрацию, при этом аппаратура шифрации и дешифрации (при «параллельной» реализации) содержит ряд однотипных модулей.

На рис. 6.19 показаны «параллельный» и «последовательный» вариаиты построения системы передачи информации по каналу связи. Входное информационное слово x0—x6 (рис. 6.19, *а*) через группу усилителей передается в канал связи. Одновременно с этим с помощью шифратора формируется группа контрольных разрядов c0—c6, которая также поступает в канал связи. Дешифратор формирует выходной код x'0—x'6, равный поданному на вход шифратора, несмотря на возможные ошибки допустимой кратности в ка-

нале связи. Устройство исправляет все одиночные ошибки в информационных разрядах x0—x6, все одиночные или двойные ошибки в контрольных разрядах c0—c6, а также часть двойных ошибок, одна из которых находится в информационных, а другая — в контрольных разрядах. (Корректирующие возможности можно увеличить.)

При использовании последовательного шифратора (рис. 6.19, *б*) в канал связи сначала поочередно передаются информационные биты x0, x1, ..., x6, а затем контрольные c0, c1, ..., c6. Дешифратор запоминает полученные информационные и контрольные биты в сдвиговых регистрах, а затем выдает исправленную (при наличии допустимых ошибок в канале) последовательность информационных битов x′0, x′1, ..., x′6 на выход.

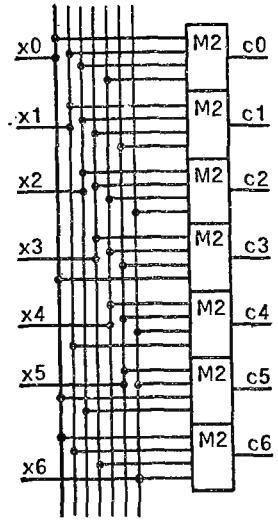


Рис. 6.20. Параллельный шифратор (см. рис. 6.19, *а*)

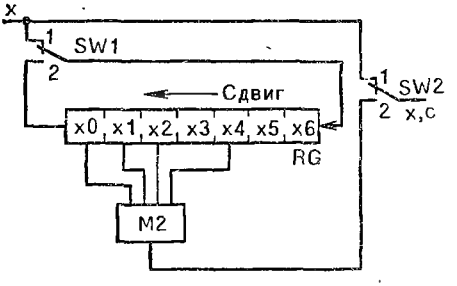


Рис. 6.21. Последовательный шифратор (см. рис. 6.19, *б*)

На рис. 6.20 и 6.21 приведены схемы параллельного и последовательного шифраторов, используемых в схемах, показанных на рис. 6.19. Параллельный шифратор (рис. 6.20) содержит семь 4-входовых элементов суммирования по модулю два (M2). В схеме последовательного шифратора (рис. 6.21) используются мультиплексоры, условно показанные в виде механических переключателей SW1, SW2, сдвиговый регистр RG и элемент суммирования по модулю два (M2). В исходном состоянии переключатели SW1, SW2 установлены в положение 1, информационные биты x поступают в канал связи и одновременно в сдвиговый регистр. После заполнения регистра оба переключателя устанавливаются в положение 2, сдвиговый регистр «замыкается в кольцо», с выхода элемента суммирования по модулю два в канал связи поступает последовательность «c» контрольных разрядов.

На рис. 6.22 показана схема параллельного дешифратора (см. рис. 6.19, *а*). Восстановленные биты x′0—x′6 формируются 5-входовыми мажоритарными элементами Maj, каждый из которых выполняет функцию голосования по большинству входных сигналов (единица на выходе формируется при наличии трех или более единиц на входах). Схемы формирования сигналов x′0—x′6 одинаковы.

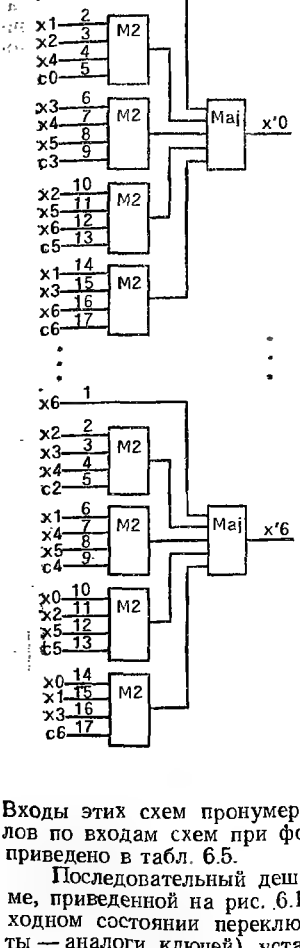


Рис. 6.22. Параллельный дешифратор (см. рис. 6.19, *а*)

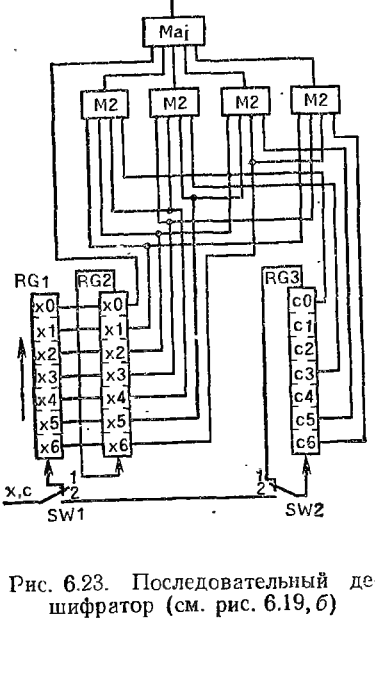


Рис. 6.23. Последовательный дешифратор (см. рис. 6.19, *б*)

Входы этих схем пронумерованы от 1 до 17, распределение сигналов по входам схем при формировании выходных сигналов x′0—x′6 приведено в табл. 6.5.

Последовательный дешифратор (рис. 6.23), используемый в схеме, приведенной на рис. 6.19, *б*, работает следующим образом. В исходном состоянии переключатели SW1 и SW2 (логические элементы — аналоги ключей) установлены в положение 1. Информационные биты x, поступающие из канала связи, запоминаются в сдвиговом регистре RG1. После заполнения регистра RG1 переключатели переводятся в положение 2, так что контрольные разряды «с» заполняют регистр RG3. Затем переключатели вновь переводятся в поло-

Таблица 6.5

Выход	Номера входов схемы (см. рис. 6.22)															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
x'0	x0	x1	x2	x4	c0	x3	x4	x5	c3	x2	x5	x6	c5	x1	x3	x6	c6
x'1	x1	x0	x2	x4	c0	x2	x3	x5	c1	x4	x5	x6	c4	x0	x3	x6	c6
x'2	x2	x0	x1	x4	c0	x1	x3	x5	c1	x3	x4	x6	c2	x0	x5	x6	c5
x'3	x3	x1	x2	x5	c1	x2	x4	x6	c2	x0	x4	x5	c3	x0	x1	x6	c6
x'4	x4	x0	x1	x2	c0	x2	x3	x6	c2	x0	x3	x5	c3	x1	x5	x6	c4
x'5	x5	x1	x2	x3	c1	x0	x3	x4	c3	x1	x4	x6	c4	x0	x2	x6	c5
x'6	x6	x2	x3	x4	c2	x1	x4	x5	c4	x0	x2	x5	c5	x0	x1	x3	c6

жение 1, информация из регистра RG1 параллельным кодом переписывается в регистр RG2. С выхода мажоритарного элемента считывается восстановленный бит x'0. Далее информация в регистрах RG2 и RG3 одновременно циклически сдвигается, а с выхода мажоритарного элемента считываются последующие восстановленные биты x'1—x'6. Регистр RG1 можно исключить и вводить информацию непосредственно в регистр RG2. При этом, однако, усложняется алгоритм работы переключателей.

В [124] приведена структура кодообразующей матрицы для построения кодов с различной способностью исправления ошибок.

6.16. Защита программы от ее прогона на «чужой» микроЭВМ, однотипной со «своей» [125, 197]

Предположим, что имеется ряд однотипных микроЭВМ, выпущенных одним и тем же заводом по одной и той же документации, с использованием одинаковых комплектующих элементов. Микропереключатели, перемычки установлены в одинаковые положения, производительность всех микроЭВМ одинакова. Задача состоит в том, чтобы программа пользователя выполнялась только на его собственной микроЭВМ и не выполнялась на любой другой однотипной микроЭВМ.

Как выявить «индивидуальность» микроЭВМ в ряду ей подобных? Оказывается, что своего рода «отпечатком пальца» микроЭВМ может служить картина разрушения тестовой информации в динамическом ОЗУ при отключении регенерации. Индивидуальность и повторяемость этой картины обусловлена индивидуальными отличиями физической структуры кристаллов однотипных микросхем динамической памяти.

В табл. 6.6 приведены результаты считывания информации из динамического ОЗУ одноплатной микроЭВМ «Электроника МС 1201.03» при отключении регенерации на различные промежутки времени (1, 2, ..., 5, 10, 15 с).

Рассматривая правый столбец таблицы, соответствующий длительному отключению регенерации (15 с и более), отмечаем, что в ячейках 0—16 и 40—56 ОЗУ устанавливается нулевая, а в ячейках 20—36 и 60—76 — единичная информация. Это связано с тем, что в БИС динамической памяти 50 % запоминающих элементов

Таблица 6.6

Адрес	Время отсутствия регенерации, с						
	1	2	3	4	5	10	15
0	177777	177777	143777	103577	003137	000000	000000
2	177777	177777	143777	103577	003115	000000	000000
4	177777	177777	177777	103577	003137	000000	000000
6	177777	177777	143777	103577	003115	000000	000000
10	177777	177777	143777	103577	003115	000000	000000
12	177777	177777	143777	103577	003115	000000	000000
14	177777	177777	177777	103577	003137	000000	000000
16	177777	177777	143777	103577	003115	000000	000000
20	000000	000000	034000	074020	174662	177777	177777
22	000000	000000	034000	074200	174662	177777	177777
24	000000	000000	034000	174200	174662	177777	177777
26	000000	000000	034000	174040	174662	177777	177777
30	000000	000000	034000	174020	174662	177777	177777
32	000000	000000	034000	174600	174662	177777	177777
34	000000	000000	034000	074200	174662	177777	177777
36	000000	014000	034000	070600	174662	177777	177777
40	177777	177777	143777	103577	003137	000000	000000
42	177777	177777	163777	103577	003137	000000	000000
44	177777	177777	147777	103177	003137	000000	000000
46	177777	177777	143777	103577	003137	000000	000000
50	177777	177777	177777	103577	003137	000006	000000
52	177777	177777	163777	103177	003135	000006	000000
54	177777	177777	147777	103577	003137	000000	000000
56	177777	177777	143777	103577	003115	000006	000000
60	000000	000000	034000	174600	174662	137773	177777
62	000000	014040	034040	174240	174662	137773	177777
64	000000	000000	034000	174000	174662	177777	177777
66	000000	014000	034000	174600	174660	177777	177777
70	000000	000000	034000	174200	174662	177777	177777
72	000000	014000	034000	174600	174662	137773	177777
74	000000	000000	034000	174600	174662	177777	177777
76	000000	014000	034000	174640	174660	137773	177777

хранят информацию в прямом коде, а остальные 50% — в обратном. (Для построения внутренней шины данных используются оба плеча триггера, в котором фиксируется входной информационный бит.) Пользователю эта особенность БИС не заметна, так как при считывании ранее проинвертированный бит вновь инвертируется, т. е. выдается в той же фазе, что и при записи. Однако при отсутствии регенерации с течением времени все запоминающие конденсаторы «затвор—исток» элементов памяти на МДП-транзисторах разряжаются и все эти элементы переходят в одинаковые состояния (условно — лог. 0). С учетом того, что каждому элементу памяти соответствует неинвертирующий или инвертирующий усилитель считывания,

при опросе всех элементов памяти БИС получим 50 % лог. 1 и 50 % лог. 0, что и наблюдается (см. столбец 15 табл. 6.6).

Если перед отключением регенерации записать в динамическое ОЗУ коды, приведенные в столбце 15 табл. 6.6, то после отключения регенерации эти коды остаются постоянными, так как в них уже «более нечему разрушаться». Все конденсаторы запоминающих элементов исходно уже разряжены. Напротив, «максимальному разрушению» подвергнется «противоположная» информация, представленная в столбце 1 табл. 6.6, поэтому она выбрана в качестве тестовой. Тестовая информация соответствует заряженному состоянию конденсаторов «затвор—исток» всех запоминающих элементов.

Из табл. 6.6 (столбец 1) следует, что по истечении 1 с после отключения регенерации тестовая информация не исказилась. Если возобновить регенерацию, а затем отключить ее уже на 2 с, то на данном образце микроЭВМ получим распределение кодов в ячейках, представленное в столбце 2 или близкое к нему. После включения регенерации, восстановления исходной тестовой информации и отключения регенерации на 3 с получим распределение кодов, близкое или совпадающее с представленным в столбце «3» таблицы и т. д.

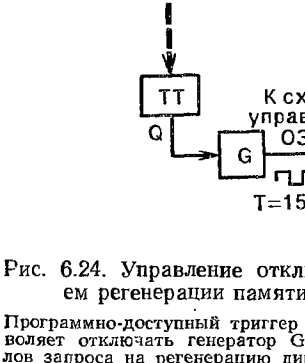


Рис. 6.24. Управление отключением регенерации памяти.

Программно-доступный триггер ТТ позволяет отключать генератор G сигналов запроса на регенерацию динамического ОЗУ. Вместо триггера может использоваться разряд выходного порта микроЭВМ

Скорость разрушения тестовой информации зависит от температуры, однако характер разрушения остается примерно постоянным: «слабые» элементы памяти теряют информацию быстрее, чем «сильные». Потеря информации может быть немонотонной. Например, в ячейке с адресом 56 (табл. 6.6) после кода 5 в младшей восьмеричной цифре формируется код 6 и лишь после этого — код 0 (столбцы 5, 10, 15).

В качестве индивидуального «отпечатка», согласно [125, 197], используется тестовая информация, разрушенная на 10 %. Бо́льшая или меньшая степень разрушения скрывает индивидуальные отличия памятей разных микроЭВМ; например, после полного разрушения тестовой информации (см. столбец 15 табл. 6.6) все микроЭВМ становятся неотличимыми друг от друга.

При получении «отпечатка» процессор, работая по программе, проводит серию экспериментов, подобных описанным ранее (см. табл. 6.6), и добивается получения массива информации, в котором присутствует 10 % ошибочных кодов. Полученный массив представляет собой экспериментально полученный «отпечаток». Он в дальнейшем прикладывается к массиву защищаемой информации и хранится вместе с ним на внешнем носителе данных, например на гибком магнитном диске.

Пользовательская программа, предусматривающая защиту информации, построена так, что после считывания «отпечатка» с магнитно-

го диска процессор сверяет этот отпечаток с экспериментально полученным. Если совпадение хорошее (например, более 90 %), то разрешается доступ к защищаемому массиву информации, в противном случае программа препятствует считыванию этого массива. Таким образом, программа способна принять решение о возможности предоставления защищаемой информации пользователю: если он работает на «своей» микроЭВМ, то доступ к информации разрешен; если пользователь работает на «чужой» микроЭВМ, то информация не считывается или даже преднамеренно уничтожается.

На рис. 6.24 показана схема, обеспечивающая программно-управляемое включение и выключение генератора G запросов регенерации. При нормальной работе микроЭВМ триггер ТТ установлен в состояние Q=1, генератор G включен, схема управления ОЗУ по запросам генератора производит перебор строк накопителей и их регенерацию с периодом 15,5 мкс. Для выключения регенерации процессор переводит программно-доступный триггер ТТ в состояние Q=0, генератор G прекращает формирование импульсов. Для возобновления регенерации процессор переводит триггер ТТ в состояние Q=1.

Если программа получения «отпечатка» находится в ПЗУ, то отключение регенерации ОЗУ не приводит к каким-либо отрицательным последствиям (за исключением того, что информация в ОЗУ теряется). Однако если программа получения «отпечатка» размещена в том же ОЗУ, которое лишается регенерации, возникает опасность искажения самой программы. Чтобы программа не искажалась, она строится таким образом, чтобы строки накопителя, в которых она размещена, циклически регенерировались самой программой. При считывании команд и обращении к операндам автоматически регенерируются соответствующие строки накопителя, так что программа сама себя «спасает» от разрушения. В тех строках, которые не затрагиваются программой, информация разрушается и эти строки можно использовать для размещения исходной тестовой информации для получения «отпечатка». Размер «отпечатка» может составлять, например, 1К 16-разрядных ячеек.

6.17. Перемещение отказавших страниц динамического ОЗУ в конечную его часть

Если при тестировании динамического ОЗУ обнаружен отказ одной или нескольких ячеек, то, используя предлагаемые схемные решения, можно переместить одну или две страницы с отказавшими ячейками в конечную область адресного пространства ОЗУ. При этом образуется непрерывная область из исправных страниц. Это позволяет упростить программное обеспечение микроЭВМ, так как при распределении памяти не нужно «обходить» неисправные страницы — они отодвинуты за пределы непрерывной рабочей области.

Предположим, что в нашем распоряжении имеется ОЗУ, содержащее 256К 16-разрядных ячеек и выполненное на 16 БИС типа К565РУ7 со структурой 256К×1 бит. Полный 18-разрядный адрес передается в БИС ОЗУ в два этапа, по сигналам RAS и CAS. По сигналу RAS передаются девять младших, затем по сигналу CAS — девять старших разрядов адреса (обычно группы адресных разрядов передаются в БИС в обратной последовательности — сначала старшая, а затем младшая). При такой передаче адреса имеется резерв времени, позволяющий без потери быстродействия ОЗУ провести пе-

ренумерацию страниц, определяемых старшими разрядами адреса, с тем чтобы «отодвинуть» ошибочные страницы в конечную часть адресного пространства ОЗУ. Преобразование старших разрядов адреса производится в то время, пока микросхема ОЗУ воспринимает младшие разряды адреса.

На рис. 6.25 показана схема передачи адреса к БИС ОЗУ с преобразованием старших разрядов. Схема позволяет присвоить одной ошибочной странице максимальный номер, равный 777_8, т. е. переместить ее в конец адресного пространства ОЗУ.

Рис. 6.25. Схема преобразования адреса для «вытеснения» ошибочной ячейки в конечную область адресного пространства

Полный 18-разрядный адрес ОЗУ проходит сквозь регистр-защелку RG_A и по сигналу L_A фиксируется в этом регистре. (Регистр-защелка с потенциальным входом синхронизации, например типа К531ИР22, в отличие от регистра с импульсным входом синхронизации типа К531ИР23, в исходном состоянии транслирует входные сигналы на выходы, что создает благоприятные условия для более быстрой передачи адреса на входы БИС ОЗУ.) Младшие разряды А0—А8 адреса через открытые сигналом ERAS усилители передаются на входы БИС ОЗУ. (Усилители входят в состав микросхемы К531ИР22, но для наглядности показаны в виде отдельного блока.)

По фронту сигнала RAS (не показан) все 16 БИС ОЗУ принимают 9 младших разрядов адреса А0—А8. После этого сигнал ERAS снимается, формируется сигнал ECAS, который открывает вторую группу усилителей для выдачи в 9-разрядную адресную шину старших разрядов адреса А'9—А'17. Эти разряды принимаются в БИС ОЗУ по сигналу CAS (не показан), а затем схема переходит в исходное состояние.

Сигналы А'9—А'17 формируются в результате поразрядного суммирования по модулю два сигналов А9—А17 и С9—С17. Значения сигналов С9—С17 вычисляются процессором по результатам начального тестирования ОЗУ и заносятся в программно-доступный ре-

гистр RG_C. В исходном состоянии в регистре RG_C записан нулевой код.

Предположим, что в результате начального тестирования после включения напряжения питания выяснилось, что на странице с номером 5 имеется неисправная ячейка памяти. В этом случае номер 777 следует преобразовать в номер 5, чтобы с точки зрения процессора ошибочная страница оказалась в конце адресного пространства ОЗУ. Для этого процессор записывает в регистр RG_C (рис. 6.25) код 772_8, полученный инвертированием кода 5. В результате код $A=777$ преобразуется в код $A'=5$ (рис. 6.26, *а*). Остальные коды

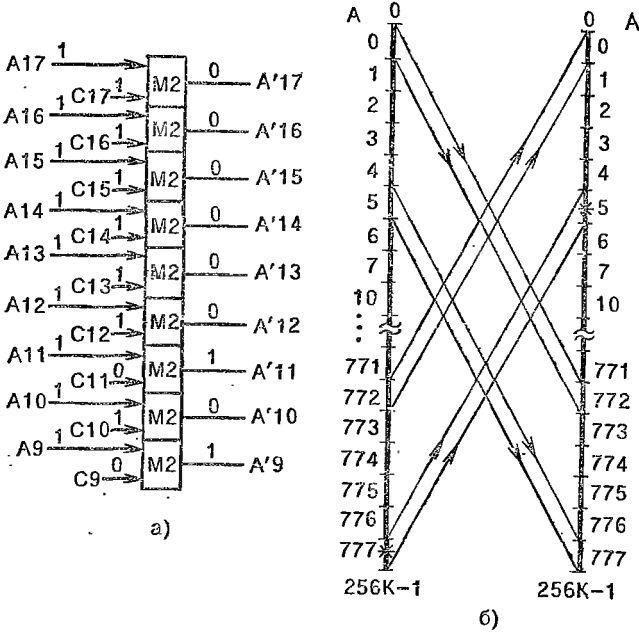


Рис. 6.26. Преобразование адреса 777××× в адрес 005××× (*а*); возникающее при таких преобразованиях перераспределение страниц в адресном пространстве (*б*)

также преобразуются (рис. 6.26, *б*), что не нарушает правильной работы ОЗУ.

Если имеются две неисправные ячейки, принадлежащие разным страницам ОЗУ, то их перемещение с помощью схемы, представленной на рис. 6.25, не всегда приводит к удовлетворительным результатам. На рис. 6.27, *а* показано преобразование номеров 5 и 7 неисправных страниц в номера 775 и 777. Страница с номером 776 оказалась изолированной от массива исправных страниц с номерами 0—774 и поэтому не используется программами. На рис. 6.27, *б* показана неблагоприятная ситуация, когда преобразование адресов

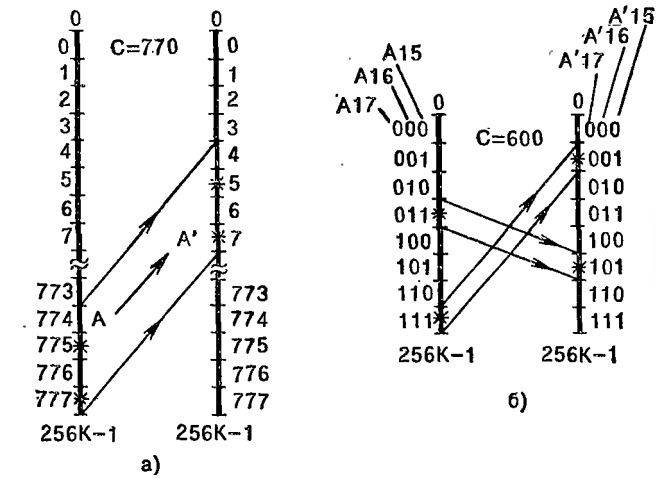


Рис. 6.27. Диаграммы перемещения двух ошибочных адресов в конечную часть адресного пространства с помощью схемы, приведенной на рис. 6.25:

а — относительно удачная попытка перемещения (осталась только одна изолированная страница); *б* — неудачная попытка, так как невозможно сдвинуть в конечную область одновременно два ошибочных адреса

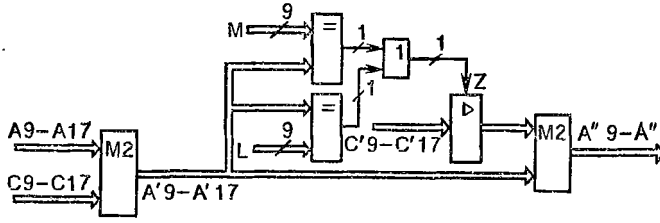


Рис. 6.28. Двухкаскадная схема преобразования адреса для «вытеснения» двух неисправных страниц ОЗУ в конечную область адресного пространства

приводит к получению непрерывного массива всего лишь из 96К ячеек (3 блока по 32К ячеек).

Для преобразования номеров X и Y двух неисправных страниц в номера 776 и 777 можно использовать схему, показанную на рис. 6.28. Ее работа поясняется рис. 6.29 (X=775, Y=4). Преобразование адреса производится в два этапа. На первом этапе номер 777 преобразуется в номер 4 (половина задачи решена). «Побочным эффектом» этого преобразования является переход номера 776 в номер 5. Поэтому на втором этапе номера 5 и 775 меняются местами (L→X, M→L), остальные номера страниц не преобразуются.

Для получения сигналов A′9—A′17 используется поразрядное суммирование старших разрядов адреса с кодом C9—C17, как и в примере, приведенном на рис. 6.25. На втором этапе преобразования адреса производится либо прямая передача кода A′9—A′17 на выходы A″9—A″17, если нет совпадения этого кода с константами M или L, вычисленными процессором, либо передача с инвертированием определенных разрядов, если сработал один из компараторов. В первой ситуации усилители закрыты сигналом Z=0 и на их выходах сформированы сигналы лог. 0. Во второй ситуации Z=1, код A′9—A′17 поразрядно суммируется по модулю два с кодом C′9—C′17, который содержит единицы в несовпадающих разрядах кодов L и M (см. кодовые комбинации, приведенные на рис. 6.29). Код C′9—

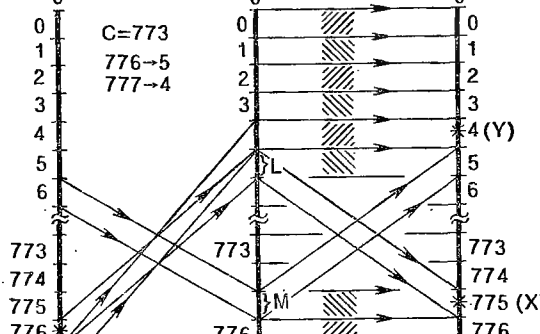


Рис. 6.29. Двойное преобразование адресного пространства схемой, приведенной на рис. 6.28

C′17 вычисляется процессором или формируется с помощью элементов суммирования по модулю два кодов L и M.

Схему при необходимости можно усложнить для «плотной упаковки» трех и более неисправных страниц.

6.18. Резервированная система из трех микроЭВМ [126]

Система (рис. 6.30) содержит три микроЭВМ, два мультиплексора MUX1, MUX2, выходные порты P1, P2, P3, преобразователи P/S1, P/S2 параллельного кода в последовательный, компаратор и блок управления F. Система предназначена для обработки информации, поступающей по одному или нескольким каналам связи (на рис. 6.30 не показаны), и выдачи данных на выход в последовательном коде.

Все три микроЭВМ работают по одинаковым программам. В ходе выполнения программы каждая микроЭВМ выдает в соответствующий порт (P1, P2, P3) очередной результат вычислений. Существенно, что при выдаче информации из порта P3 она инвертируется. Два других порта (P1 и P2) выдают информацию в прямом коде.

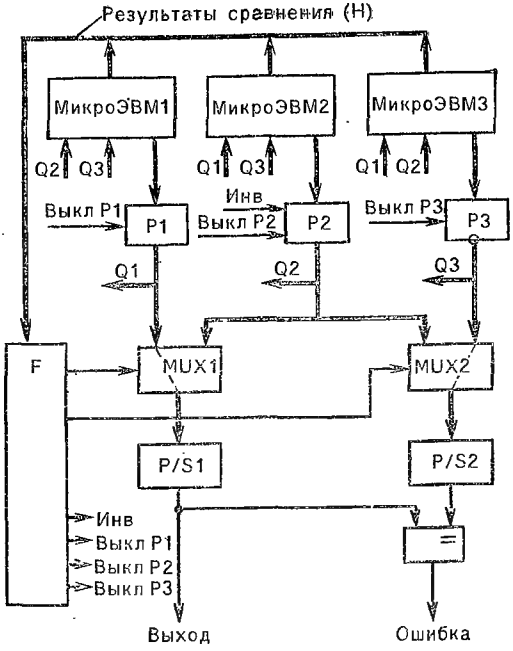


Рис. 6.30. Резервированная система.

Информация, выдаваемая одной из микроЭВМ, представлена инверсным кодом, что позволяет дополнительно контролировать аппаратуру переключения каналов

Каждая микроЭВМ получает информацию от двух соседних, так как входы микроЭВМ1 соединены с шинами Q2, Q3, входы микроЭВМ2 — с шинами Q1, Q3, входы микроЭВМ3 — с шинами Q1, Q2. После заполнения всех трех портов P1, P2, P3 одинаковой информацией (с учетом инверсии сигналов на шине Q3 по отношению к сигналам на шинах Q1, Q2) все три микроЭВМ сравнивают полученные результаты (с учетом инверсии). При правильной работе всех микроЭВМ каждая из них убеждается в том, что соседи получили аналогичный результат вычислений. (Цепи взаимной синхронизации не рассматриваются как не имеющие прямого отношения к существу предлагаемого решения.)

При правильной работе всех микроЭВМ мультиплексоры MUX1 и MUX2 остаются в исходном состоянии, при котором информация передается через них в направлениях, показанных штриховыми линиями. Информация с шины Q1 проходит через мультиплексор MUX1 и преобразователь P/S1 и далее в последовательном коде поступает на выход системы. Инвертированная информация с шины Q3 проходит через мультиплексор MUX2, преобразователь P/S2 и поразрядно сравнивается с выходной информацией системы. При правильной работе системы информация, поступающая на входы компаратора, противофазна, так что на выходе компаратора сформирован сигнал лог. 0.

Если одна из микроЭВМ обнаружила, что полученный ею результат отличается от результатов, полученных каждым из соседей, или если эта микроЭВМ вообще неспособна работать, то на шине результатов сравнения (H) устанавливается определенный код, который распознаётся блоком управления F.

В зависимости от номера i отказавшей микроЭВМ блок F формирует потенциальный сигнал ВыклP_i, который «необратимо» отключает соответствующий порт. Отказ микроЭВМ2 не приводит к изменению направлений передачи информации через мультиплексоры MUX1 и MUX2. При отказе микроЭВМ1 блок управления F переключает мультиплексор MUX1, так что в дальнейшем информация с шины Q2 проходит через мультиплексор MUX1 и далее через преобразователь P/S1 на выход системы (вместо микроЭВМ1 к выходу системы подключается микроЭВМ2). Состояние мультиплексора MUX2 не изменяется.

При отказе микроЭВМ3 переключается мультиплексор MUX2 и функции этой микроЭВМ выполняет микроЭВМ2. Но так как порт P3 инвертировал передаваемую информацию, порт P2 также должен ее инвертировать. Поэтому одновременно с выключением порта P3 из блока управления F на режимный вход порта P2 подается потенциальный сигнал Инв, который переводит этот порт в режим инвертирования передаваемой информации.

Благодаря использованию прямых и инвертированных кодов и их сравнению с помощью компаратора, обеспечивается более полный контроль работы системы. Например, если при правильной работе всех микроЭВМ ошибочно переключились оба мультиплексора одновременно или только мультиплексор MUX2, то информация на входы компаратора поступает не в противофазе, а в фазе и на выходе компаратора формируется сигнал лог. 1 (Ошибка). Аналогично будет обнаружено ошибочное переключение мультиплексора MUX1, если порт P2 работает в режиме инвертирования передаваемой информации, ошибочное формирование сигнала Инв при отказе микроЭВМ1 и проч.

6.19. Точный или приближенный поиск желаемой последовательности символов в проходящем потоке символов [91]

Задача быстрого поиска некоторой заранее известной последовательности символов может быть решена с использованием схемы, показанной на рис. 6.31 (приведено пять состояний схемы). Схема содержит сдвиговый регистр RG, блок компараторов и блок счетчиков CT. В схему поступает поток байт, например, с магнитного

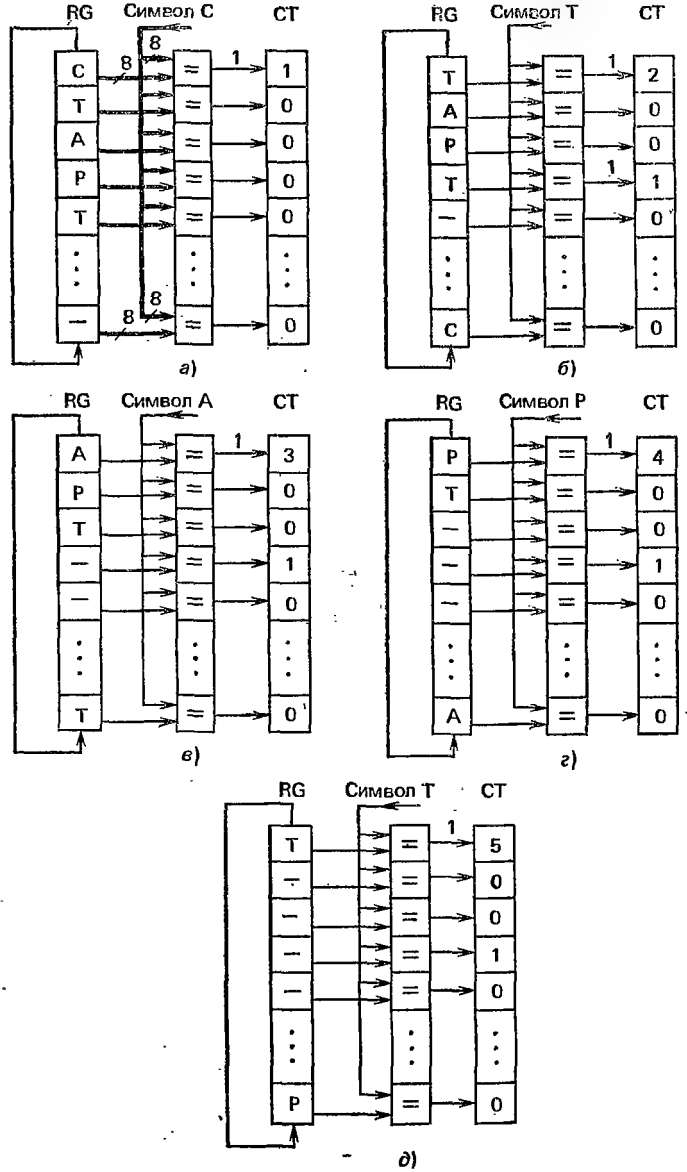


Рис. 6.31. Этапы поиска слова «старт»

диска или с магнитной ленты. Задача состоит в том, чтобы опознать в проходящем потоке нужную комбинацию символов, например фамилию, год и место рождения человека, сведения о котором должны быть выданы на печать.

Предположим, что требуется отыскать в проходящем потоке слово «старт». Каждый символ этого слова представлен одним байтом (8 бит). Искомое слово загружается в сдвиговый регистр и «бесконечно вращается» в нем, каждый раз сдвигаясь на одну символьную позицию. Момент сдвига совпадает с моментом поступления извне очередного символа.

Рассмотрим ситуацию, показанную на рис. 6.31, *а*. Слово «старт» оказалось в верхней части кольцевого сдвигового регистра RG (это не имеет принципиального значения и принято только для наглядности рисунка) к моменту поступления извне символа С. При этом верхний компаратор регистрирует совпадение символа, поступающего извне, с символом, поступившим из верхнего символьного разряда (8 бит) сдвигового регистра RG. На выходе этого компаратора формируется единица, которая регистрируется верхним счетчиком, первоначально установленным в 0. После этого слово «старт» вновь сдвигается на одну символьную позицию, а извне, как предполагается, поступает символ Т.

Этот символ совпадает с содержимым двух символьных разрядов регистра RG, так как в слове «старт» две буквы Т. Поэтому в верхнем счетчике будет зарегистрирована еще одна единица (рис. 6.31, *б*), а в четвертом сверху счетчике накапливается одна единица. Далее на вход устройства поступает символ А (рис. 6.31, *в*), в результате в верхнем счетчике накапливаются три единицы. При поступлении на вход символа Р верхний счетчик вновь оказывается «в центре событий» и прибавляет единицу (рис. 6.31, *г*), наконец при поступлении последнего символа Т в верхнем счетчике накапливается пять единиц, по числу букв в слове (рис. 6.31, *д*).

Таким образом, если хотя бы в одном счетчике зарегистрировано пять единиц, то задача считается выполненной. Можно констатировать, что «только что» в проходящем потоке «промелькнуло» нужное сочетание символов. Дальнейшее поведение поисковой системы может быть самым разнообразным — нас сейчас интересует именно его начало.

Описанный процесс точного поиска может не дать положительных результатов, например, при «опечатках», допущенных при вводе информации оператором, при опознании иероглифов, которые каждый человек пишет с индивидуальными отклонениями от эталона, и т. д. Для приближенного поиска, как видно из схемы, нужно снизить требования к числу накопленных единиц или, другими словами, снизить «порог» при анализе состояния счетчиков на одну, две или более единиц в расчете на «опечатки».

Если не принять специальных мер по периодическому сбросу счетчиков, то в них будет накапливаться «шум» из-за случайных совпадений кодов символов. Поэтому практическая схема реализации данного метода содержит «перемещающийся по кольцу» бит сброса счетчиков, который «пристроен» вслед за эталонным словом. Кроме того, предусматриваются следующие режимы управления устройством:

1. При несовпадении кодов и ненулевом результате в счетчике из него вычитается единица. Это — своего рода «штраф» за несов-

падение, который увеличивает «контрастность» картины распределения кодов в счетчиках.

2. При несовпадении накопленный в счетчике результат остается без изменения, как показано на рис. 6.31.

3. При несовпадении счетчик устанавливается в 0. Этот режим соответствует точному поиску слова.

Описанный метод, таким образом, позволяет осуществлять точный или приближенный поиск нужных данных на максимальной скорости потока.

Иное решение задачи точного поиска связано с использованием схемы [92] (рис. 6.32), напоминающей микропрограммное устройство управления. Предположим, что во входном потоке данных нужно опознать следующее сообщение (каждый символ представлен 6-разрядным кодом): —JOHN—DOE—AT6374N—. Процесс опознания поясняется табл. 6.7.

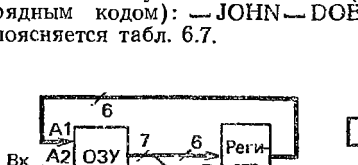


Рис. 6.32. Устройство для поиска заранее известного сочетания символов.

A1 — старшие разряды

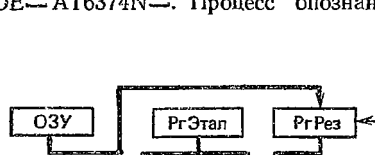


Рис. 6.33. Устройство для анализа информации в ОЗУ

В исходном состоянии в регистре записан нулевой код. Этот код записывается в регистр всякий раз, когда сравнение оказалось неудачным. Предположим, что очередной код, поступивший на входы Вх, соответствует символу «пробел» (—). Это код 001100. Полный 12-разрядный адрес ОЗУ, таким образом, равен 000000001100, или в десятичном представлении 12. Как видно из табл. 6.7, в ячейке 12 записан код 000001. Старший разряд (Вых) равен нулю. Код D= =000001 записывается в регистр и служит для задания шести старших разрядов следующего адреса. Предположим далее, что вслед за символом «—» поступил символ J (код 010110). С учетом полученных ранее шести старших разрядов полный адрес равен $000001010110_2=86_{10}$ (см. строку 2 табл. 6.7). В этой ячейке памяти записан код 000010, который перепишется в регистр для задания шести старших разрядов следующего адреса и т. д.

Если искомое сообщение найдено, то на конечном этапе поиска (см. последнюю строку табл. 6.7) в старшем разряде слова данных, выбранного из ОЗУ, появится единица. Во всех ячейках памяти, не указанных в табл. 6.7 (их число равно 4К—18), записаны нули, поэтому любой «срыв» с цепочки вновь приведет в начальную точку поиска, в которой содержимое регистра RG равно нулю и ожидается появление символа «—». Устройство может быть запрограммировано на одновременный поиск нескольких сообщений, а также на

Таблица 6.7

Символ	A1	A2	Ячейка	D	Выход
—	000000	001100	12	000001	0
J	000001	010110	86	000010	0
O	000010	011011	155	000011	0
H	000011	010100	222	000100	0
N	000100	011010	282	000101	0
—	000101	001100	332	000110	0
D	000110	010000	400	000111	0
O	000111	011011	475	001000	0
E	001000	010001	529	001001	0
—	001001	001100	588	001010	0
A	001010	001101	653	001011	0
T	001011	100000	736	001100	0
6	001100	000110	774	001101	0
3	001101	000011	835	001110	0
7	001110	000111	903	001111	0
4	001111	000100	964	010000	0
N	010000	011010	1050	010001	0
—	010001	001100	1100	000001	1

выполнение арифметических операций, проверку соотношений между числами, проверку попадания чисел в заданные границы и т. д. В силу этого предложенное устройство названо «ассоциативным процессором».

6.20. Поиск в ОЗУ слова, минимально отличающегося от эталонного слова [93]

На рис. 6.33 представлена схема, предназначенная для поиска слова, наиболее близкого к предъявленному. Массив кодовых слов, среди которых ищется нужное слово, предварительно записывается в ОЗУ. Эталонное слово, по отношению к которому ищется наиболее близкое, записывается в регистр РгЭтал. В регистре результата РгРез постепенно или с первой же попытки сравнения накапливается «наилучший» код. В исходном состоянии в регистре РгРез записано инвертированное эталонное слово (максимально «удаленное» от эталона).

Массив слов последовательно выдвигается из ОЗУ, каждое слово сравнивается с эталонным с помощью компаратора К1. На выходе компаратора К1 формируется число X, равное числу разрядов, в которых произошло несовпадение. Например, при сравнении кодов 01011011 и 01011000 будет выявлено несовпадение в двух разрядах, поэтому будет сформировано число X=2. Компаратор К2 работает аналогично. Компаратор К3 выдает на выход импульс только в том случае, если X<Y, т. е. если очередной код с выхода ОЗУ более «похож» на эталонный код, чем тот, который взят в качестве наилучшего приближения на предыдущих шагах и хранится в регистре

РгРез. Поэтому в регистр РгРез принимается новый код по сигналу с выхода компаратора К3. Если код, поступивший из ОЗУ, оказался «худшим», чем код в регистре РгРез (X≥Y), то он игнорируется.

Таким образом, «прогнав» через устройство все содержимое ОЗУ, можно отыскать код, максимально близкий к заданному. При обнаружении точного совпадения поиск прекращается.

6.21. Устройство для защиты информации в коллективной памяти от посторонних пользователей [95]

Для засекречивания и рассекречивания информации используются соответственно кодирующая и декодирующая программируемые логические матрицы (см. § 1.4) ПЛМ1 и ПЛМ2, как показано на рис. 6.34. Пример кодировки ПЛМ1:

$$D'0=\overline{A0}\cdot\overline{A1}\cdot\overline{A2}\cdot D7 \vee \overline{A0}\cdot\overline{A1}\cdot A2\cdot D1 \ldots A0\cdot\overline{A1}\cdot A2\cdot D3 \vee A0\cdot A1\cdot D2;$$

$$D'1=\overline{A0}\cdot\overline{A1}\cdot\overline{A2}\cdot D4 \vee \overline{A0}\cdot\overline{A1}\cdot A2\cdot D2 \ldots A0\cdot\overline{A1}\cdot A2\cdot D5 \vee A0\cdot A1\cdot \overline{D6};$$

$$D'2=\overline{A0}\cdot\overline{A1}\cdot\overline{A2}\cdot D6 \vee \overline{A0}\cdot\overline{A1}\cdot A2\cdot D7 \ldots A0\cdot\overline{A1}\cdot A2\cdot \overline{D0} \vee A0\cdot A1\cdot D1;$$

. . . . . .

$$D'7=\overline{A0}\cdot\overline{A1}\cdot\overline{A2}\cdot \overline{D3} \vee \overline{A0}\cdot\overline{A1}\cdot A2\cdot D4 \ldots A0\cdot\overline{A1}\cdot A2\cdot D6 \vee A0\cdot A1\cdot D5.$$

Смысл этих записей таков. Если, например, в трех младших разрядах адреса при записи (Зп) присутствует код 000_2 ($\overline{A0}\cdot\overline{A1}\cdot\overline{A2}$), то производится такое «перепутывание» линий шины данных: линия D'0 подключается к линии D7, линия D'1 — к линии D4, D'2 — к D6 и т. д. Линия D'7 подключается к линии D3 через внутренний инвертор ПЛМ1. Если в младших разрядах адреса обнаружена комбинация 11× (A0·A1 независимо от значения A2), то D'0=D2, D'1=$\overline{D6}$ и т. д.

Декодирующая ПЛМ2 содержит «обратную» кодировку, которую легко получить, зная исходную. Она производит восстановление информации, которая в «перепутанном» виде хранится в коллективной памяти, при чтении (Чт). Посторонний пользователь, не знающий ко-

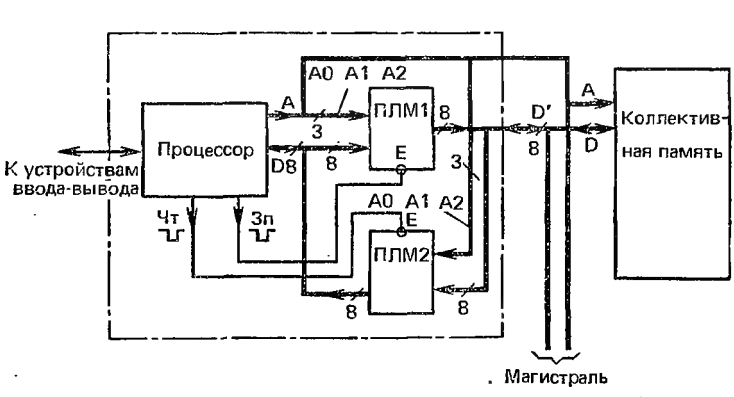


Рис. 6.34. Использование ПЛМ для защиты информации

дировки ПЛМ1, практически не имеет шансов за приемлемое время расшифровать информацию, зашифрованную таким способом (возможны гораздо более сложные уравнения кодирования). Разумеется, доступ к контактам ПЛМ подлежит запрету, для чего процессор вместе с обеими ПЛМ выполняется, например, в виде моноблока, залитого эпоксидной смолой.

6.22. Повышение обнаруживающей способности системы контроля, использующей код Хэмминга [148]

При построении ОЗУ с коррекцией одиночных и обнаружением двойных ошибок в хранимом слове данных обычно используют код Хэмминга. Для коррекции одиночных и обнаружения двойных ошибок в 32-разрядном слове данных необходим 7-разрядный код Хэмминга, получаемый с помощью формирователя (рис. 6.35, *а*) при записи входного слова данных в ОЗУ. В примере, приведенном на рис. 6.35, *а*, полное 39-разрядное слово (32 информационных и 7 контрольных разрядов) разбито на три части, каждая часть (13 разрядов) хранится в одной из плат (1, 2, 3). При считывании данных схема проверки и коррекции кода исправляет одиночную ошибку путем инвертирования ошибочного бита при использовании одного из выходных элементов Исключающее ИЛИ (Mod 2). Формирователь и схема проверки могут быть выполнены на одних и тех же элементах.

При обнаружении двойной ошибки схема проверки и коррекции считанного кода формирует сигнал Некорректируемая ошибка, который может, например, вызывать прерывание работы центрального процессора. Этот же сигнал вырабатывается и при обнаружении некоторых (но не всех!) ошибок тройной или большей кратности. Проблема как раз и заключается в том, что групповые ошибки могут не обнаруживаться, но именно такие ошибки возникают при отказе общих цепей управления и адресации, охватывающих несколько разрядов. Число таких общих цепей управления и адресации в плате (рис. 6.35, *б*) достаточно велико: эти цепи сигналов $\overline{RAS}_0$, $\overline{RAS}_1$ (выбор строки в одной из «линеек» БИС ОЗУ), CAS (выбор столбца, сигнал поступает во все БИС ОЗУ), WE (разрешение записи, сигнал поступает во все БИС ОЗУ) и сигналов А0—А8 для передачи групп старших и младших разрядов полного 18-разрядного адреса во все БИС ОЗУ типа К565РУ7 со структурой 256К×1 бит. Индивидуальными являются лишь сигналы входных и выходных данных. На рис. 6.35, *б* в каждом разряде использованы две БИС ОЗУ с объединенными информационными входами и выходами; общая информационная емкость двух БИС при таком включении составляет 512К бит.

Чтобы повысить обнаруживающую способность системы контроля, предлагается использовать общие цепи управления и адресации так, чтобы каждое ответвление (через усилитель) от любой такой цепи обслуживало не более двух разрядов хранимого слова (рис. 6.36). Например, отказ выхода усилителя D3, транслирующего адресный сигнал А0 только к двум парам разрядных БИС ОЗУ, приведет к появлению ошибки в двух разрядах, а система контроля по Хэммингу гарантирует обнаружение двойных ошибок. Поэтому отказ выхода усилителя D3 будет обнаружен. Для обнаружения отказа выхода усилителя D1 используется схема Mod 2 контроля ад-

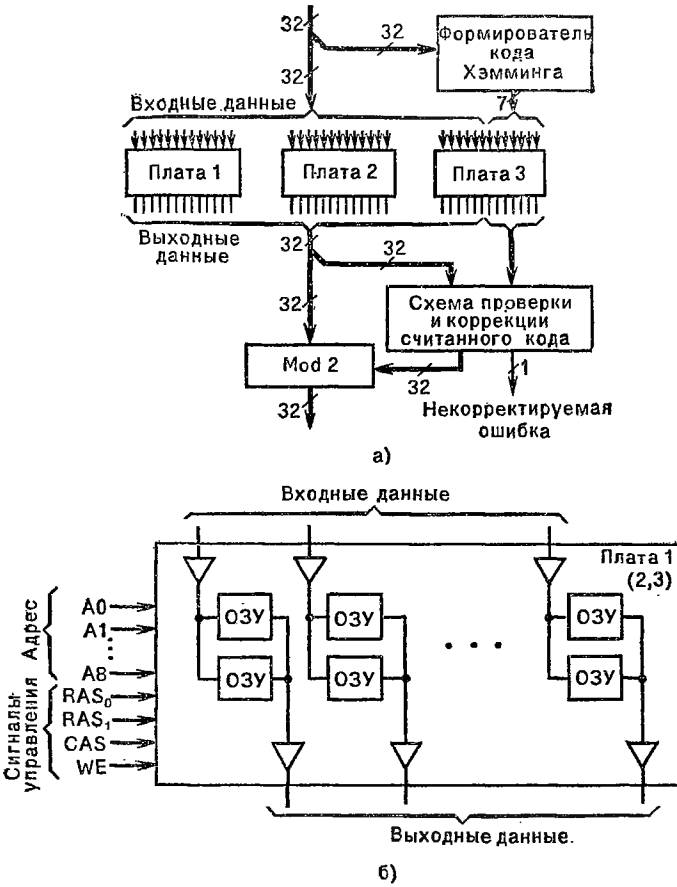


Рис. 6.35. ОЗУ с коррекцией информации по Хэммингу:

а — функциональная схема; *б* — структура платы

ресных сигналов по четности (или нечетности), на которую поступают сигналы A0—A8 и дополнительно введенный внешний сигнал контрольного разряда (не показан), дополняющий общее число лог. 1 в 9-разрядном коде адреса до четного или нечетного (это обычная схема контроля, но в данном случае она дополняет контроль по Хэммингу). При нарушении условия четности (нечетности) числа единиц в адресном коде вырабатывается сигнал ошибки ERR (схемы стробирования для простоты не показаны).

Распределение общих сигналов управления по плате производится аналогично. Каждый вторичный усилитель обслуживает один или два разряда, в последнем случае отказ выхода усилителя при-

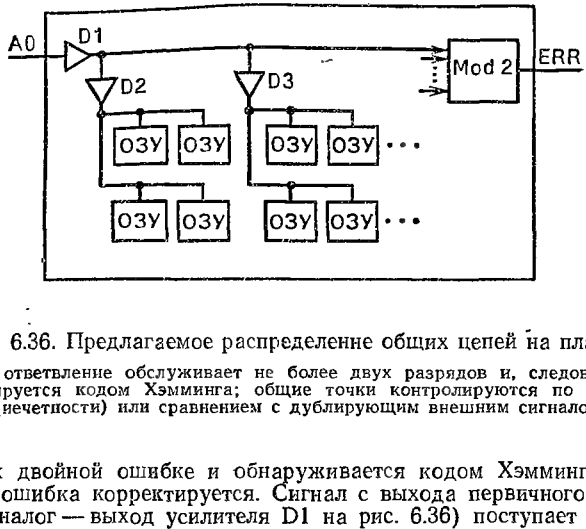


Рис. 6.36. Предлагаемое распределенне общих цепей на плате.

Каждое ответвление обслуживает не более двух разрядов и, следовательно, контролируется кодом Хэмминга; общие точки контролируются по четности (нечетности) или сравнением с дублирующим внешним сигналом

водит к двойной ошибке и обнаруживается кодом Хэмминга, одиночная ошибка корректируется. Сигнал с выхода первичного усилителя (аналог — выход усилителя D1 на рис. 6.36) поступает на схему сравнения с дополнительно введенным дублированным сигналом управления, поступающим на плату извне, так что отказ выхода первичного усилителя также обнаруживается при несовпадении основного и дублирующего сигналов.

6.23. Повышение вероятности обнаружения ошибок путем их размножения [150, 164]

Рассматриваются ошибки при передаче информации по каналу свизи или (н) при ее хранении в памяти. Ошибки могут быть вызваны, например, помехами по цепям электропитания, попаданием посторонних частиц под головку считывания — записи магнитофона, пролетом α-частиц через кристаллы динамической памяти и т. п.

Простейший способ обнаружения ошибок — контрольное суммирование кодов, образующих некоторый массив, и сравнение полученной суммы с эталоном, который входит в этот массив. На рис. 6.37, *а* массив состоит из множества информационных байтов $A_1, A_2, ..., A_N$ и байта S контрольной суммы. При записи массива в память или при пересылке в канал связи передаваемые байты поразрядно суммируются между собой по модулю два, в результате накапливается контрольная сумма S, которая передается вместе с массивом информационных байтов, а при считывании или при приеме из канала связи служит эталоном для сравнения.

Недостаток этого способа — плохая обнаруживающая способность по отношению к групповым ошибкам, так как ошибки внутри группы могут компенсировать друг друга. Например, если бит 6 в байте A_3 ошибочно перешел из 0 в 1, а этот же бит в байте A_4 — из 1 в 0, то эта двойная ошибка не будет обнаружена, так как считанная контрольная сумма S будет соответствовать вычисленной.

Для улучшения обнаруживающей способности по отношению

к групповым ошибкам в соседних байтах используют суммирование с расслоением (рис. 6.37, *б*), при котором контрольная сумма S_A является результатом поразрядного суммирования по модулю два байтов A_i, S_B — байтов B_i и S_C — байтов C_i. Однако это также не дает достаточно полной гарантии обнаружения ошибок.

Предлагается более эффективный способ, при котором имеющиеся ошибки размножаются и проникают во все разряды считываемого массива байтов, что способствует повышению вероятности их последующего обнаружения. Чтобы понять смысл этого способа, рассмотрим сначала, как ошибку можно размножить.

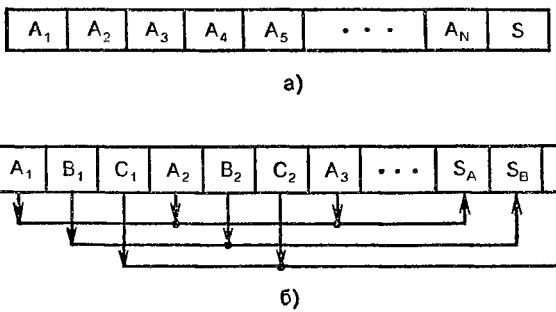
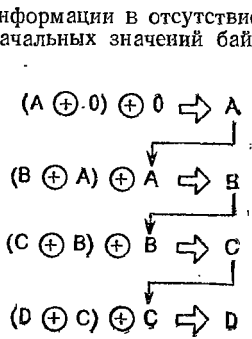


Рис. 6.37. Контрольное суммирование:
а — без расслоения; *б* — с расслоением

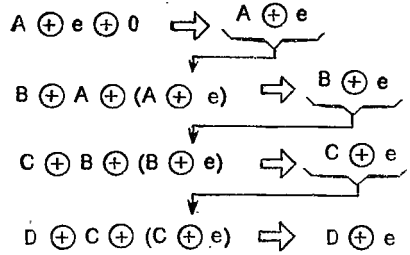
Предположим, что нужно запомнить в памяти или передать в канал связи массив, состоящий из байтов A, B, C, D и т. д. В отличие от обычной последовательной передачи можно передавать такую последовательность байтов: $A \oplus 0$, $B \oplus A$, $C \oplus B$, $D \oplus C$ и т. д. Иными словами, каждый последующий байт поразрядно суммируется по модулю два с предыдущим и результат суммирования передается в память или канал связи. Для байта A предыдущим считается нулевой.

При считывании информации в отсутствие ошибок производится восстановление первоначальных значений байтов A, B, C, D и т. д.:



Смысл этой записи таков. Сначала (первая строка) считывается первый байт $A \oplus 0$ и поразрядно суммируется по модулю два с нулевым кодом. В результате получаем байт А. Затем (вторая строка) считывается второй байт $B \oplus A$ и суммируется с байтом А, полученным ранее. В результате получаем байт В, который используется для восстановления байта С и т. д.

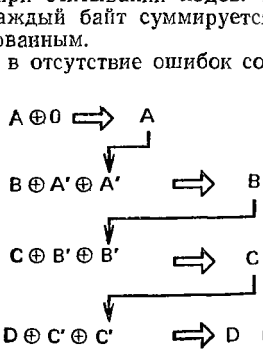
Если первый байт считан с ошибкой, т. е. вместо кода $A \oplus 0$ считан код $A \oplus e$, где e — байт ошибки, содержащий одну или несколько единиц, то эта ошибка распространится и на остальные байты:



Если ошибка была в одном разряде первого считанного байта, то она теперь присутствует в том же разряде всех считанных байтов. Остается «разбросать» полученную цепочку ошибок по разным разрядам кодов, попутно увеличивая число ошибок, тогда увеличится вероятность их обнаружения при окончательном контрольном суммировании восстановленных байтов и сравнении полученной суммы с эталонной (см. рис. 6.37, *а*).

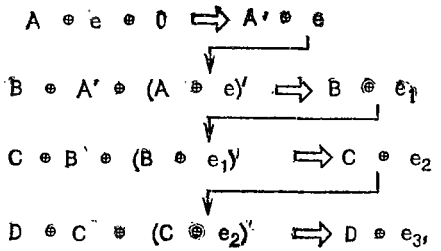
В общем виде предлагаемый способ предусматривает следующую последовательность действий. При подготовке массива для его передачи в канал связи или в память формируется цепочка байтов: A, $B \oplus A'$, $C \oplus B'$, $D \oplus C'$ и т. д. Запись вида X' соответствует некоторому (вполне определенному) преобразованию кода X; это преобразование должно по возможности обеспечивать перенос информации из разряда в разряд, чтобы ошибка «блуждала» по разрядам восстанавливаемых при считывании кодов. Таким образом, в предлагаемом способе каждый байт суммируется с предыдущим, но на этот раз — преобразованным.

При считывании в отсутствие ошибок события развиваются так:



Первый считанный байт A преобразуется в байт A′ и затем используется для восстановления байта B, который, в свою очередь, преобразуется в байт B′ и т. д.

При наличии ошибки e в первом считанном байте



где байты ошибок e_1, e_2, e_3 и т. д. при правильном выборе вида преобразования кодов являются псевдослучайными кодами. Таким образом, ошибка размножается и «поражает» случайные разряды всех считанных кодов. При контрольном суммировании восстановленных байтов фактически используются случайные байты, поэтому вероятность получения правильной контрольной суммы снижается и ошибка с высокой вероятностью будет обнаружена.

Рассмотренный метод положен в основу системы обнаружения ошибок, в которой используется как поразрядное суммирование по модулю два, так и арифметическое суммирование кодов с циклическим переносом.

Предположим, что нужно передать в память массив кодов, показанный на рис. 6.38, *а*. Байт A имеет численное значение 15_8, байт

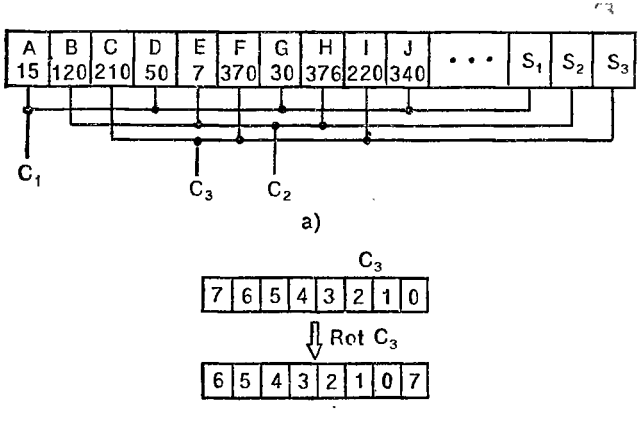


Рис. 6.38. Фрагменты системы с размножением ошибок:

а — массив данных; *б* — схема циклического сдвига

$B - 120_8$ и т. д. Байты контрольных сумм S_1—S_3 вычисляются арифметическим суммированием соответствующих групп байтов, показанных на рисунке. При суммировании перенос «с» из старшего разряда арифметически складывается с младшим разрядом суммы. Переменные C_1—C_3 введены для обозначения текущих значений контрольных сумм S_1—S_3. В процессе кодирования (при записи) и декодирования (при чтении) код C_3 циклически сдвигается влево на один разряд (рис. 6.38, б). Эта операция условно обозначена символом $RotC_3$. В исходном состоянии (как при чтении, так и при записи) $C_1=C_2=C_3=0$. В скобках приведены результаты выполнения операций.

1. Кодирование информации для ее передачи в память.

Rot C_3 (0)

$C_1 = C_1 + A + c$ (15)

$A \oplus C_3 \rightarrow$ Память (15)

$C_2 = C_2 + B + c$ (120)

$B \oplus C_1 \rightarrow$ Память (135)

$C_3 = C_3 + C + c$ (210)

$C \oplus C_2 \rightarrow$ Память (330)

Rot C_3 (21)

$C_1 = C_1 + D + c$ (65)

$D \oplus C_3 \rightarrow$ Память (71)

$C_2 = C_2 + E + c$ (127)

$E \oplus C_1 \rightarrow$ Память (62)

$C_3 = C_3 + F + c$ (12)

$F \oplus C_2 \rightarrow$ Память (257)

Rot C_3 (24)

$C_1 = C_1 + G + c$ (115)

$G \oplus C_3 \rightarrow$ Память (14)

$C_2 = C_2 + H + c$ (126)

$H \oplus C_1 \rightarrow$ Память (263)

$C_3 = C_3 + I + c$ (244)

$I \oplus C_2 \rightarrow$ Память (306)

Rot C_3 (111)

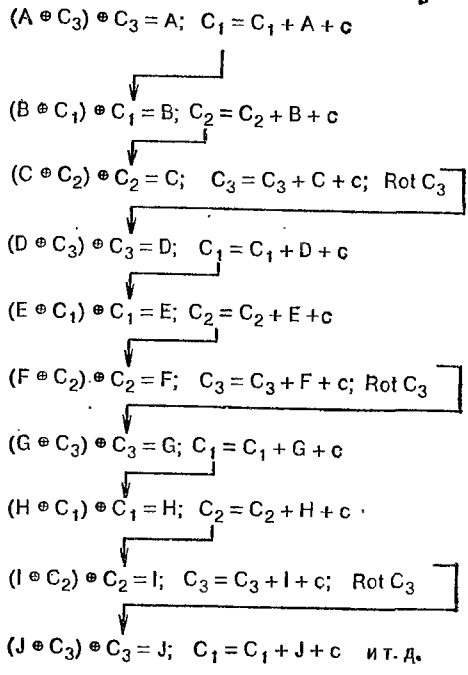
$C_1 = C_1 + J + c$ (56)

$J \oplus C_3 \rightarrow$ Память (251) и т. д.

В результате кодирования в память записывается не ряд кодов, показанных на рис. 6.38, *а* (15, 120, 210, 50, 7 и т. д.), а следующий ряд: 15, 135, 330, 71, 62, 257, 14, 263, 306, 251 и т. д.

2. Декодирование информации при ее считывании из памяти. Восстанавливаются все приведенные на рис. 6.38, *а* коды (15, 120,

210, 50, 7 и т. д.). На каждом шаге декодирования восстанавливается очередной исходный код и вычисляется константа для восстановления последующего кода. Если из памяти считан хотя бы один код с ошибочным битом, то «рушится» вся последующая цепочка вычислений, а ошибка множится, проникает во все разряды восстанавливаемых кодов. Поэтому контрольное суммирование полученного массива «случайных» кодов вряд ли даст результат, совпадающий с эталоном (S_1—S_3). Далее приведены правила декодирования.



6.24. Схема защиты информации в ОЗУ от ошибочной записи (по материалам [152])

В схеме, приведенной на рис. 6.39, центральный процессор ЦП имеет возможность беспрепятственного считывания кодов из ОЗУ. Однако чтобы записать код в ОЗУ, он должен «попросить на это разрешения» установкой триггера ТТ в состояние $\overline{Q}=0$, после чего ему предоставляется возможность однократной записи кода.

После включения напряжения питания или при выполнении команды начальной установки на линии Сброс формируется сигнал лог. 0, который переводит триггер ТТ в состояние $\overline{Q}=1$, соответст-

вующее наличию защиты ОЗУ от записи. Выполнение команды чтения информации из ОЗУ сопровождается формированием на адресной шине адреса ячейки памяти. Истинность адреса при чтении подтверждается сигналом RD=0, который проходит на вход выбора кристалла ВК дешифратора DC1 старших разрядов адреса ОЗУ. Сигнал RD=0 запирает элемент И—НЕ по верхнему входу, в результате сигнал $\overline{Q}=1$ запрета доступа к ОЗУ блокируется и не препятствует формированию сигнала E=1 запуска ОЗУ. Считанный из ОЗУ код по шине данных передается в процессор, затем все магистральные сигналы снимаются.

При попытке записи сигнал $\overline{Q}=1$ препятствует формированию сигнала E=1 запуска ОЗУ. Чтобы получить разрешение на запись,

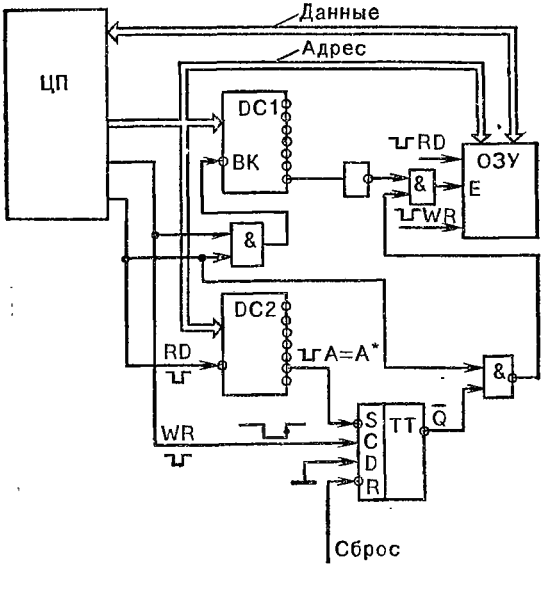


Рис. 6.39. Схема защиты ОЗУ от ошибочной записи

процессор выполняет команду чтения информации из ячейки памяти с адресом A*, который не принадлежит адресам ОЗУ и распознаётся дешифратором DC2 (дешифратор DC1 остается в пассивном состоянии). В результате триггер ТТ переходит в состояние $\overline{Q}=0$, элемент И—НЕ закрывается по нижнему входу. Последующая команда записи выполняется беспрепятственно. Сигнал WR=0 подтверждает истинность адреса (и данных) при записи; по окончании этого сигнала триггер ТТ автоматически устанавливается в запрещающее состояние, и чтобы вновь записать в ОЗУ код, потребуется получение нового разрешения.

6.25. Контроль выполнения программы с использованием таймера [157]

Если программа управления объектом выполняется циклически, то ее работу можно проконтролнровать с нспользованием таймера, выполняющего функции «будильника». В начале (или в любом другом месте) каждого цикла таймер программно устанавливается в исходное состояние н начинает отсчет времени. При правильной работе микроЭВМ будильник не срабатывает, так как время выполнения программного цикла меньше интервала времени, установленного в будильнике. Если в результате сбоя или отказа программа зациклилась на каком-либо участке, не содержащем команду начальной установки таймера, то срабатывает будильник и в процессор посылается сигнал сброса, который приводит микроЭВМ в исходное состояние с последующим перезапуском (или перезагрузкой и перезапуском) программы.

Недостаток такого способа контроля состоит в том, что если программа ошибочно зациклилась на участке, содержащем команду начальной установки таймера, то будильник не сработает, ошибка не будет обнаружена, перезапуска процессора не произойдет и микроЭВМ останется в неработоспособном состоянии.

Чтобы предотвратить такие ситуации, в программу вводятся несколько команд начальной установки таймера, последовательность выполнения которых известна. Например, при правильной работе программы первая команда начальной установки таймера должна выбираться из ячейки памяти с адресом 2500, вторая — из ячейки 15004, третья—десятая — из ячейки 2000 (внутренний цикл, выполняемый 8 раз), одиннадцатая — из ячейки 4400, двенадцатая (новое вхождение в цнкл) — из ячейки 2500 и т. д. Если с помощью внешней аппаратуры следить за последовательностью прохождения адресов команд начальной установки таймера, то можно повысить достоверность контроля.

Предлагаемая схема рис. 6.40 подключается к магистрали микроЭВМ; перед пуском рабочей программы она настраивается на слежение за заданной последовательностью адресов команд начальной установки таймера. Для этого ожидаемая последовательность адресных кодов записывается в последовательные ячейки памяти ОЗУ2; в нулевую ячейку записывается код 2500, в первую — 15004, во вторую—девятую — коды 2000, в десятую — 4400, в одиннадцатую — 2500 (новый цикл), в двенадцатую — 15004 и т. д.

В исходном состоянии счетчик СТ установлен в 0, на выходах регистра RG2 присутствует код 2500. После пуска рабочей программы процессор, как обычно, выбирает из ОЗУ1 последовательность команд и выполняет их. Таймер, первоначально установленный в исходное состояние, ведет отсчет времени, вычитая единицы из своего содержимого с некоторой частотой. Адреса извлекаемых из ОЗУ1 команд временно запоминаются в регистре RG1. При выполнении команды начальной установки таймера формируется сигнал RTR=1, который разрешает работу компаратора и опрашивает результат сравнения. При правильной работе процессора первая команда начальной установки таймера извлекается из ячейки ОЗУ1 с адресом 2500, т. е. компаратор формирует на выходе сигнал сравнения (лог. 1), который проходит через элемент И, устанавливает таймер

в исходное состояние и вызывает прибавление единицы к содержимому счетчика СТ. Теперь в регистре RG2 хранится код 15004 — адрес следующей ожидаемой команды начальной установки таймера, а таймер вновь начинает отсчет времени.

При выполнении следующей команды начальной установки таймера, если ее адрес равен 15004, вновь срабатывает компаратор, счетчик СТ увеличивает содержимое на единицу, таймер возвращается в исходное состояние и т. д. При нарушении правильной последовательности адресов команд начальной установки таймера компаратор не срабатывает и по истечении заданного интервала времени таймер формирует сигнал общего сброса микроЭВМ, по которому производится ее перезапуск.

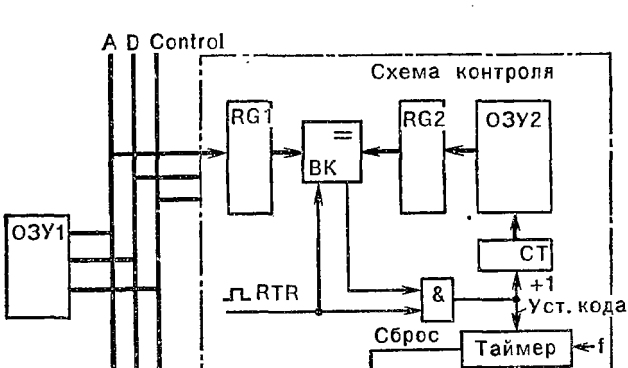


Рис. 6.40. Схема контроля микроЭВМ

Показаны основные элементы и связи

Счетчик СТ устанавливается в 0 по сигналу общего сброса, а также (автоматически) при заполнении его до определенного уровня; возможен программный сброс счетчика в начале каждого цикла выполнения рабочей программы.

6.26. Контроль передачи адресных сигналов к микросхемам памяти [160]

При построении блоков быстродействующей памяти большого объема и (или) большой разрядности необходимо передавать адресные сигналы одновременно в несколько модулей H_i, в каждом из которых размещен ряд микросхем ОЗУ или ПЗУ. Предлагаемая схема распределения адресных сигналов между модулями H_i позволяет исключить лишние ретрансляторы и обеспечить контроль передачи адреса к микросхемам памяти (рис. 6.41).

Обращение к блоку памяти производится с использованием шины DADD или CADD, по которой передается адрес операнда и ад-

рес команды. Выбор шины производится с помощью мультиплексоров MUX, включенных параллельно. Выбор нужных модулей памяти H_i осуществляется с использованием дешифратора старших разрядов адреса (от A14 и выше), который не показан с целью упрощения рисунка. Каждый из четырех «групповых» мультиплексоров,

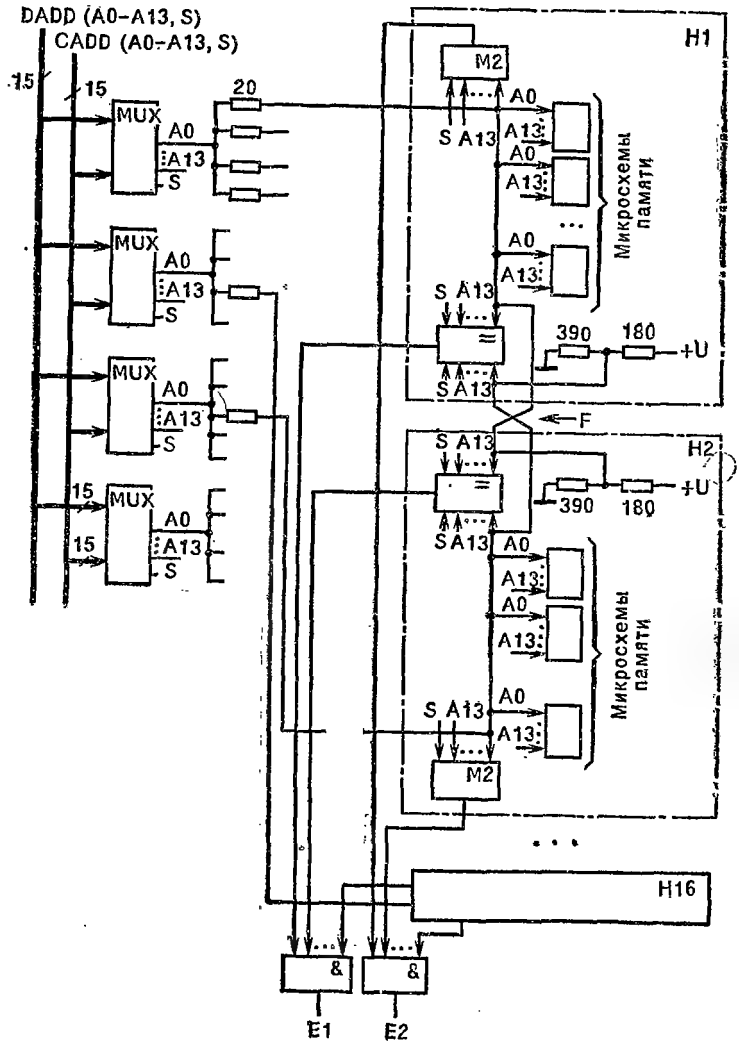


Рис. 6.41. Схема передачи адресных сигналов к микросхемам памяти

приведенных на рисунке, состоит из 15 «разрядных» мультиплексоров для индивидуальной передачи адресных сигналов A0, A1, A2, ..., A13 и контрольного разряда S, дополняющего число единиц в адресе до нечетного. На рис. 6.41 показана передача одного разряда адреса A0.

К выходу каждого разрядного мультиплексора (их число равно 60), обладающего низким выходным сопротивлением, подключены 4 резистора сопротивлением 20 Ом, которые служат для подавления колебательных процессов в кабельных линиях связи при передаче фронтов сигналов. Используются плоские кабели или их «печатные» аналоги. Каждая кабельная линия связи экранирована от соседних и согласована со стороны приемников включением резисторного делителя напряжения с эквивалентным сопротивлением, равным волновому сопротивлению линии (см. гл. 4).

Модули H1 и H2, H3 и H4, ..., H15 и H16 образуют пары, объединенные перекрестными связями F линий передачи адресных и контрольного разрядов. К каждой паре модулей одноименные сигналы (в данном примере — A0) поступают с выходов разных мультиплексоров. В каждой паре модулей с помощью компараторов производится поразрядное сравнение своих и соседних адресных сигналов. При правильной работе сигналы должны попарно совпадать, при этом на выходах компараторов формируются лог. 1. Если нет совпадения хотя бы в одной паре разрядов, на выходе компаратора формируется лог. 0.

Элементы суммирования по модулю два (M2) позволяют обнаруживать ошибки передачи адреса, связанные с нарушением условия нечетности общего числа единиц в 15-разрядном коде A0—A13, S. При правильной работе блока памяти на выходах элементов M2 сформированы лог. 1, при обнаружении ошибки — лог. 0.

Результаты работы схем контроля суммируются элементами И. При правильной работе блока памяти E1=E2=1. При обнаружении ошибки по крайней мере один из этих сигналов переходит в 0.

6.27. Контроль передачи сигналов из микроЭВМ к управляемому объекту [166]

Предлагаемая схема (рис. 6.42, *а*) позволяет контролировать наличие тока в обмотке исполнительного реле и при обнаружении неправильной работы выполнить аварийное отключение управляемого объекта.

В исходном состоянии микроЭВМ работает по некоторой программе, в ее выходном порте присутствует сигнал d, равный 0 или 1. Если d=0 (напряжение низкого уровня), то светодиод оптрона V1 излучает свет, транзистор этого оптрона насыщен, ток от положительного полюса источника напряжения U2 протекает через транзистор оптрона V1, светодиод оптрона V2 и обмотку исполнительного реле к отрицательному полюсу этого источника (источники напряжения U1 (5 В) и U2 (24 В) гальванически развязаны оптронами). Транзистор оптрона V2 насыщен, поэтому h=1, на выходе элемента Исключающее ИЛИ (M2) сформирован сигнал $h \oplus d = 1$, сигнал прерывания процессора микроЭВМ отсутствует (f=0). К обмотке защитного реле (на рисунке не показано) поступает напряжение низкого уровня, по этой обмотке протекает ток, прекращение

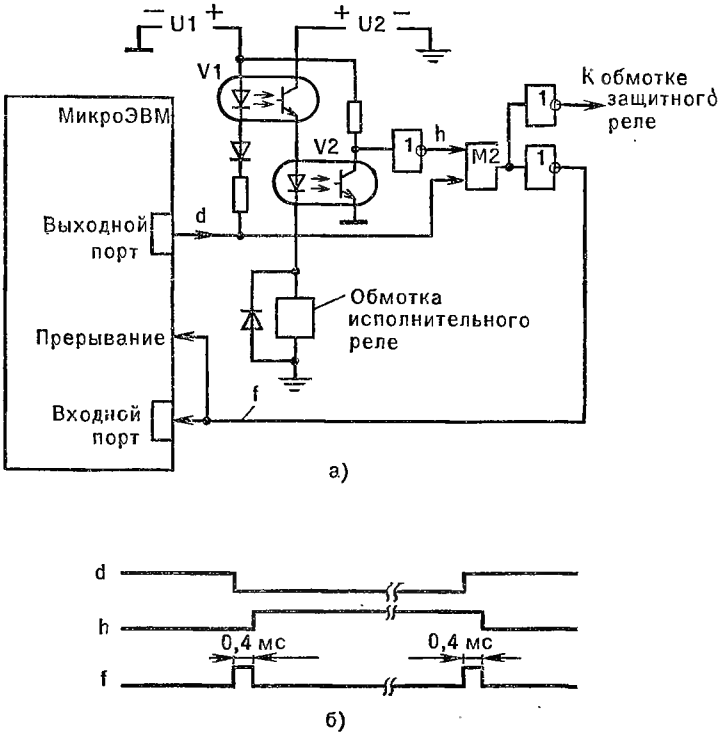


Рис. 6.42. Схема контроля подачи тока в обмотку исполнительного реле *(а)* и временны́е диаграммы ее работы *(б)*

которого при неправильной работе системы вызывает аварийное отключение управляемого объекта.

Если $d=1$, то ток через обмотку исполнительного реле не протекает, транзистор оптрона V2 выключен, $h=0$, $h \oplus d=1$, $f=0$, защитное реле включено, работа управляемого объекта разрешена.

В зависимости от состояния исполнительного реле (включено—выключено) управляемый объект работает в том или ином режиме. Если используются несколько разрядов выходного порта микроЭВМ и несколько исполнительных реле, то схема, показанная на рис. 6.42, *а*, «размножается», при этом сигналы с выходов элементов Исключающее ИЛИ объединяются дополнительной схемой И для получения обобщенного сигнала f и сигнала управления защитным реле.

В моменты изменения сигнала d сигнал f кратковременно принимает значение лог 1 из-за инерционности оптронов V1 и V2, см. рис. 6.42, *б*. Длительность импульсов неодинакова, но не превышает 0,4 мс. Наличие этих импульсов на входе прерывания

микроЭВМ при изменении сигнала d процессором подтверждает правильность отработки сигналов, передаваемых к объекту. Защитное реле не реагирует на кратковременные импульсы выключения тока через обмотку (шунтированную диодом, как и обмотка исполнительного реле), поэтому управляемый объект работает в нормальном режиме. Аварийное отключение объекта с помощью защитного реле

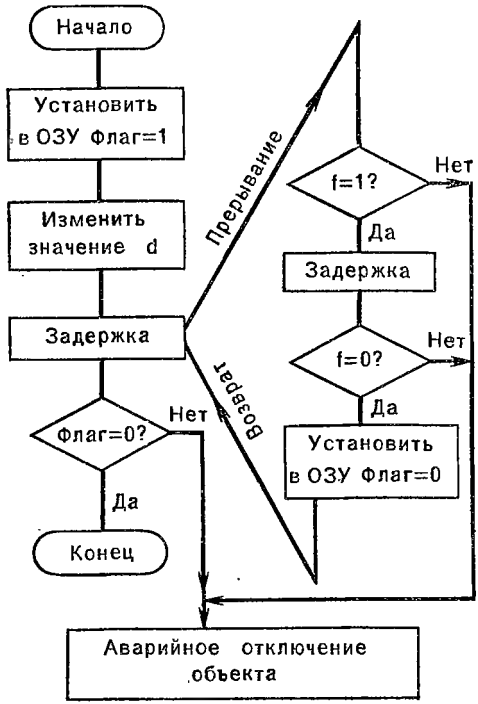


Рис. 6.43. Блок-схема подпрограммы изменения состояния исполнительного реле (см. рис. 6.42, *а*)

производится в том случае, когда на выходе элемента Исключающее ИЛИ сигнал лог. 0 поддерживается в течение времени, достаточного для срабатывания этого реле. Отметим, что это не единственная (и даже вспомогательная) мера защиты от ошибок в системе; микроЭВМ при обнаружении ошибки программными средствами также производит аварийное отключение объекта, используя выделенные для этого разряды выходного порта и соответствующую внешнюю аппаратуру (на рисунке не показана).

При необходимости изменить значение сигнала d процессор микроЭВМ обращается к подпрограмме (рис. 6.43), в которой предусмотрены следующие действия.

В одну из ячеек ОЗУ процессор записывает определенный код, например равный 111...1 (Флаг=1), затем меняет значение сигнала d (копия этого сигнала может храниться в ОЗУ) и переходит к выполнению одной-двух «пустых» команд (задержка). При изменении значения сигнала d формируется импульсный сигнал f (см. рис. 6.42, *б*), который вызывает прерывание выполнения подпрограммы. Прерывающая программа анализирует состояние сигнала f. При правильной работе системы сигнал f к моменту проверки по-прежнему равен 1 (длительность импульса 0,4 мс, а проверка производится через несколько микросекунд после его формирования). Если f=0, то это значение означает, что прерывание вызвано кратковременным импульсом (например, из-за неправильной работы микроЭВМ с выходным портом, при несанкционированной выдаче в него последовательности сигналов). В этом случае объект отключается, чтобы не возникла аварийная ситуация.

После проверки условия f=1 процессор выполняет ряд «пустых» циклов, чтобы выдержать паузу (большую 0,4 мс), достаточную для окончания импульса f (см. рис. 6.42, *б*). При последующей проверке должно выполниться условие f=0, в противном случае констатируется неправильная работа системы и управляемый объект отключается. При выполнении условия f=0 процессор устанавливает в 0 флаговую ячейку ОЗУ, подтверждая правильность формирования импульса f. (Перед установкой в 0 флаговой ячейки можно проверить, была ли она предварительно установлена в состояние 111...1, чтобы усилить контроль.) Далее производится выход из прерывающей программы и возврат к выполнению подпрограммы. Процессор проверяет состояние флаговой ячейки ОЗУ и принимает окончательное решение о возможности продолжения работы. Если Флаг=0, то производится выход из подпрограммы, в противном случае — аварийное отключение объекта.

6.28. Контроль данных по четности с повышенной вероятностью обнаружения групповых ошибок [167, 168]

При передаче или хранении слова данных с контролем по четности (нечетности) используется контрольный разряд, дополняющий общее число единиц в слове до четного (нечетного). Например, при передаче в информационные линии канала связи кода 01001100 в линию контрольного разряда передается 1, если используется контроль по четности, так как число единиц в передаваемом коде равно трем (нечетно). При приеме информации из канала связи подсчитывается общее число принятых единиц; в данном примере в отсутствие ошибок будут получены три информационные и одна единица в контрольном разряде, общее число единиц окажется четным.

При возникновении одиночной ошибки приемное устройство обнаруживает нарушение условия четности общего числа принятых единиц; однако если при передаче исказились 2, 4, 6 и т. д. бит, то ошибки компенсируют друг друга и не обнаруживаются.

Групповые ошибки могут возникнуть из-за отказа микросхемы, обслуживающей несколько разрядов. Чтобы обнаружить такие ошибки, предлагается увеличить число контрольных разрядов и распределить их так, чтобы каждый из них отвечал за правильность передачи бит, относящихся к разным микросхемам (рис. 6.44).

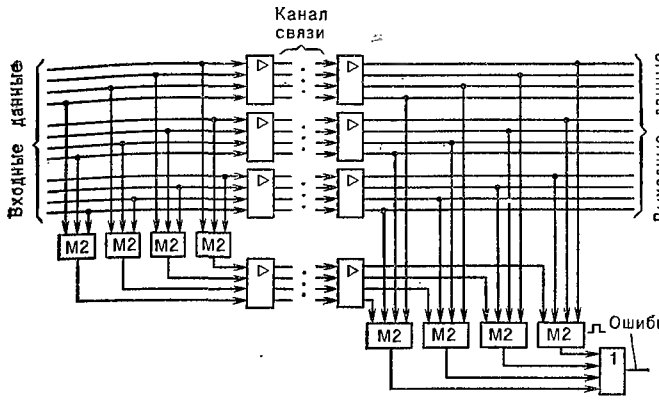


Рис. 6.44. Канал связи с контролем передаваемой информации по четности.

Обнаруживаются групповые ошибки, связанные с отказом 4-разрядной мгкросхемы передатчика или приемника

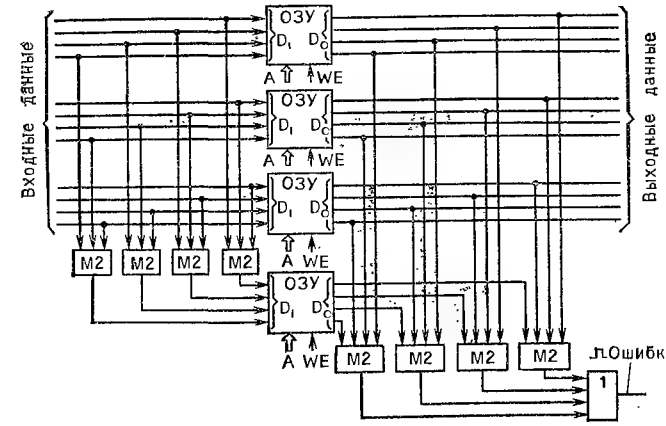


Рис. 6.45. ОЗУ с контролем хранимой информации по четности.

Обнаруживаются групповые ошибки, связаиные с отказом 4-разрядной микросхемы памяти

Каждый входной элемент Исключающее ИЛИ (М2) получает информацию с входов разных микросхем-передатчиков, а каждый выходной элемент М2 проверяет правильность полученной группы бит. При наличии одиночной ошибки на выходе одного из выходных элементов М2 формируется сигнал лог. 1, который проходит на вы-

ход элемента ИЛИ. Ошибка кратности 2, 3 и 4, возникающая из-за неисправности одной микросхемы передатчика или приемника, обнаруживается одновременно несколькими выходными элементами М2.

Аналогичный способ включения элементов контроля применим также при построении блоков памяти. В схеме на рис. 6.45 обнаруживается отказ любой микросхемы статического ОЗУ, хотя каждая из них обслуживает 4 разряда слова данных и при неисправности этой микросхемы могут исказиться сразу все 4 разряда. При записи 12-разрядного слова в три микросхемы ОЗУ формируется группа из четырех контрольных разрядов, которая запоминается в четвертой микросхеме. После считывания слова с контрольными разрядами последние используются для проверки групп бит, хранившихся в разных микросхемах памяти.

6.29. Контроль информации, считываемой из ОЗУ, без снижения его быстродействия [169]

Для обнаружения ошибок, которые могут возникнуть в хранимой в ОЗУ информации, используют схемы контроля, например по нечетности (четности). В считанном из ОЗУ слове данных, сопровождаемом контрольным разрядом, подсчитывается общее число единиц. При отсутствии ошибок это число нечетно, при наличии одиночной ошибки — четно (если принят контроль по нечетности). Подсчет числа единиц производится с помощью многовходовой схемы Исключающее ИЛИ (М2), которая вносит задержку в работу системы.

Предлагается контролировать считанный код не в текущем, а в последующем такте, что позволяет «скрыть» задержку схемы подсчета числа единиц (рис. 6.46) и не замедлять темп считывания из ОЗУ последовательности кодов. (Сама схема подсчета числа единиц может быть «медленной», ее задержка может достигать задержки считывания кода из ОЗУ.) Фактически реализован конвейерный

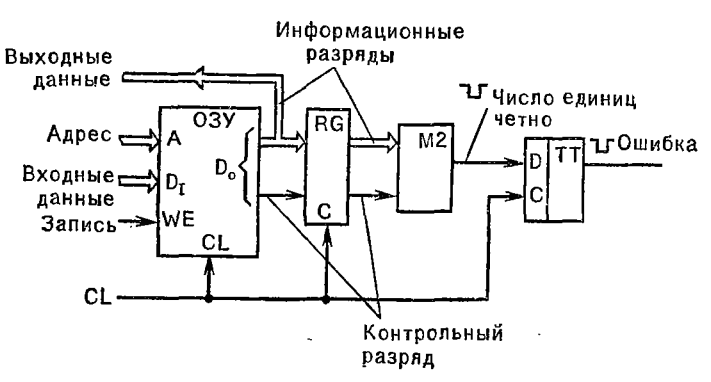


Рис. 6.46. Конвейерная схема контроля считанного из ОЗУ кода.

Результат проверки формируется с задержкой на такт по отношению к моменту появления полезной информации

режим обработки информации: пока производится считывание i-го слова из ОЗУ, (i—1)-е слово анализируется на четность (нечетность). В момент записи считанного (i-го) слова в выходной регистр RG (по фронту сигнала CL) триггер ТТ принимает с D-входа результат анализа (i—1)-го слова, а ОЗУ приступает к считыванию (i+1)-го слова (каждому слову соответствует свой адрес A). Информация об ошибке, таким образом, поступает на выход схемы с опозданием на такт, но «лучше поздно, чем никогда».

6.30. Восстановление информации, поступающей в микроЭВМ из нескольких каналов связи [173]

При передаче одинаковой цифровой информации одновременно по нескольким каналам связи, в каждом из которых возможно появление ошибок, возникает задача ее восстановления приемной станцией. В примере, приведенном на рис. 6.47, источником информации является маломощный радиопередатчик, установленный на движу-

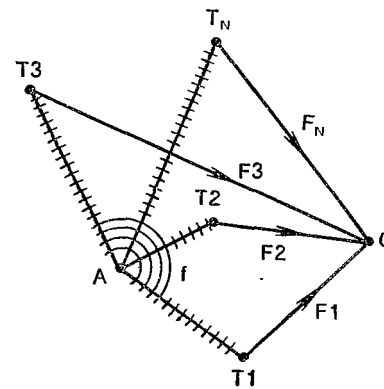


Рис. 6.47. Схема передачи цифровой информации по радиоканалам нз движущегося по городу автомобиля A в приемную станцию C.

f — частота работы автомобильного радиопередатчика, F_i — частота работы передатчика ретранслятора T_i

щемся по городу автомобиле A. Сигналы, излучаемые передатчиком на несущей частоте f, принимаются ретрансляторами T_i, которые расположены на возвышенностях в черте города. Принятые сигналы излучаются направленными антеннами ретрансляторов T_i на несущих частотах F_i в сторону приемной станции C.

Приемная станция C (рис. 6.48) содержит радиоприемники R_i, настроенные на частоты F_i. Сигналы X_i на выходах приемников R_i представлены напряжениями ТТЛ-уровней; в этих сигналах данные «смешаны» с синхроимпульсами (как, например, в коде «Манчестер-II», см. гл. 3). С помощью схем S_i производится выделение из этой смеси синхроимпульсов. Контроллеры K_i преобразуют принимаемые последовательные коды в параллельные, которые считываются процессором по магистрали.

Процессор считывает информацию из контроллеров K_i по мере их готовности к выдаче сформированных параллельных кодов. Неодновременность формирования кодов может быть вызвана разными

задержками распространения радиоволн на дистанциях A—T_i—C (см. рис. 6.47), а также разными внутренними задержками ретрансляторов T_i и приемников R_i. Далее процессор сопоставляет одноименные биты, полученные из разных каналов, и если не все они одинаковы, то принимает решение о том, каким должен быть результирующий (восстановленный) бит — 0 или 1. При такой обработке каждого бита число ошибок в восстановленном массиве информации, как правило, уменьшается по сравнению со средним числом ошибок, проникающих в каждый канал. Восстановленная информация через преобразователь P/S параллельного кода в последовательный передается в центральную ЭВМ (на рис. 6.48 не показана).

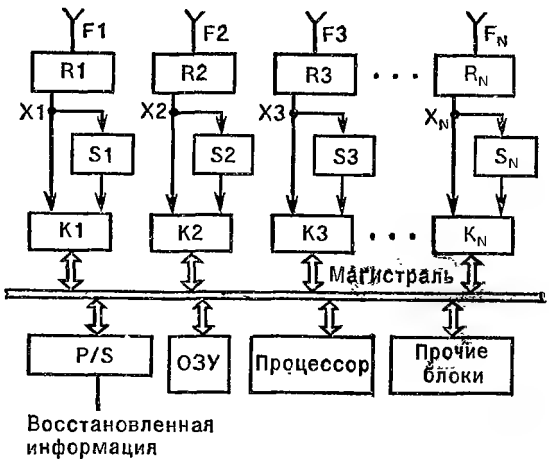


Рис. 6.48. Схема приемной станции C (см. рис. 6.47).

R_i — приемник радиосигналов частоты F_i; S_i — схема выделения сигналов синхронизации передаваемых битов; K_i — контроллер-преобразователь последовательного кода в параллельный; P/S — контроллер-преобразователь параллельного кода в последовательный

Особенность системы рис. 6.47 заключается в том, что качество сигналов, передаваемых по разным каналам, оказывается разным. Если автомобиль A находится в зоне прямой видимости ретранслятора T2, то сигнал по второму каналу, вероятнее всего, будет менее всех остальных подвержен искажениям. Однако при движении автомобиля ситуация может быстро измениться: лучший канал превращается в худший, и наоборот; все каналы могут стать «средними» и т. п. Мажоритарный отбор сигналов (по принципу голосования по большинству) применим к последней ситуации, когда среди каналов нет явного лидера, одиако если таковой имеется (один «прав», остальные, быть может, и «неправы» в своем большинстве), мажоритарный отбор приведет к тому, что восстановленная информация будет содержать больше ошибок, чем поступающая по луч-

шему каналу. Поэтому при наличии явного лидера вместо мажоритарного отбора следует провести выбор лучшего канала и пользоваться передаваемой по нему информацией.

Лучший канал обычно выбирают аналоговым способом, например сравнением мощностей сигналов, принятых разными приемниками. Однако такой способ не обеспечивает быструю реакцию на смену лидера; кроме того, радиосигнал может быть мощным, но не лучшим по качеству, например при наличии радиопомех на данной частоте. Таким образом, возникает задача построения алгоритма обработки сигналов с улучшенной восстанавливающей способностью.

Предлагаемый алгоритм (рис. 6.49) позволяет в зависимости от ситуации передавать на выход системы либо мажоритарную информацию, либо информацию, принятую из лучшего канала. Выбор того или иного режима работы производится по результатам текущего сравнения качества информации в лучшем и худшем каналах. Если интенсивность появления ошибок в худшем и лучшем каналах отличается на величину, не превышающую некоторый порог, то информация восстанавливается по мажоритарному принципу; в противном случае на выход системы передается информация из лучшего канала.

При всей логической обоснованности данного алгоритма остаются пока неясными вопросы: как вычислить интенсивность появления ошибок в каналах при отсутствии эталона, т. е. с чем сравнивать информацию, полученную из каналов, и как выбрать значение порога?

В предлагаемом алгоритме в качестве эталонной выбирается информация, полученная по мажоритарному принципу, т. е. «за неимением лучшего» эталон вычисляется на основе ненадежных данных и, следовательно, сам является ненадежным. Тем не менее приведенные далее результаты моделирования [173] подтверждают правомерность такого выбора эталона.

Предположим, что в трехканальной системе истинная интенсивность E_1 появления ошибок в первом канале составляет 0,1 %, а в двух других (E_2 и E_3) по 5 %. Это означает, что в среднем из каждой тысячи принятых бит в первом канале будет один, а в двух других — по 50 ошибочных. Интенсивность появления ошибок в информации, формируемой мажоритарным отбором, как показывает моделирование, составляет 0,26 %. Используя мажоритарно-восстановленную информацию в качестве эталона, получаем (сравнением массивов), что интенсивность E_1^* появления ошибок в первом канале равна 0,34 %, а в двух других (E_2^* и E_3^*) — по 4,75 %. Таким образом, из-за несовершенства эталона значение E_1^*, полученное экспериментально, завышено по сравнению с истинным (но никому не известным) значением E_1, а значения E_2^* и E_3^* — занижены по сравнению с E_2 и E_3. Тем не менее полученные оценки правильно отражают истинную ситуацию. Разность между полученными интенсивностями появления ошибок в худшем и лучшем каналах составляет $E_3^* - E_1^* = 4{,}75 - 0{,}34 \approx 4{,}41$ %. Если эта разность превышает заданный порог Q (а именно так должно и быть при правильном выборе порога), то в качестве выходной используется информация из первого (лучшего) канала, а второй и третий не исключаются из рассмотрения, хотя служат только для получения эталона, по качеству уступающего информации из первого канала.

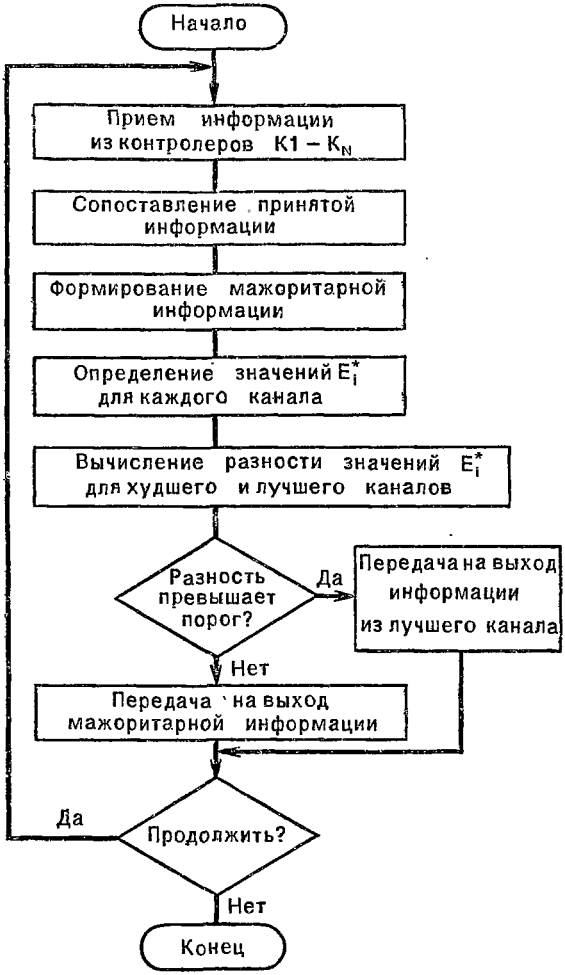


Рис. 6.49. Блок-схема алгоритма программы восстановления информации

Предположим теперь, что истинная интенсивность появления ошибок в каждом канале составляет $E_1=E_2=E_3=5\,\%$. Интенсивность появления ошибок в информации, формируемой мажоритарным отбором, составит при этом 0,725 %. Используя мажоритарно-восстановлениую информацию в качестве эталона, получаем, что

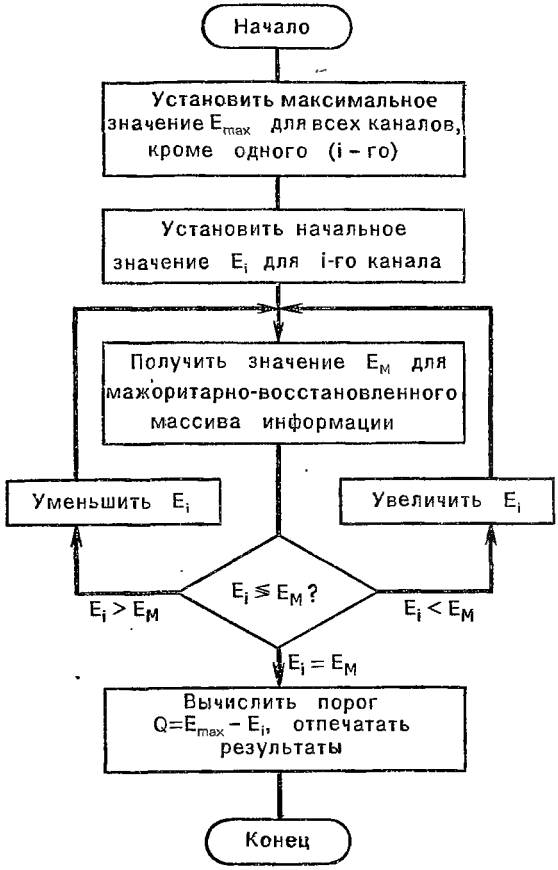


Рис. 6.50. Блок-схема алгоритма программы моделирования системы

интенсивность появления ошибок в каждом канале составляет $E_1^* = E_2^* = E_3^* = 4{,}75\ \%$. Разность интенсивностей равна 0 и так как она меньше заданного порога Q, то в качестве выходной информации принимается результат мажоритарной обработки сигналов из разных каналов. (Отметим, что, если число каналов четно и при голосовании не возникает большинства у той и другой стороны, выходной бит формируется случайным выбором между 0 и 1.)

Пороги Q для различных ситуаций могут выбираться экспериментально при опытной эксплуатации системы, однако такой путь является слишком трудоемким, по крайней мере если отсутствуют исходные расчетные данные, которые можно получить при модели-

рования системы. Моделирующая программа (рис. 6.50) позволяет провести серию вычислительных экспериментов по определению порогов. В качестве исходных данных задается наихудшая (наиболее высокая) ожидаемая интенсивность появления ошибок E_{max} во всех каналах, кроме одного (i-го), для которого выбирается некоторое начальное значение $E_i \leqslant E_{max}$. Далее формируются одинаковые массивы истинной информации, соответствующие разным каналам. В каждом массиве случайным образом искажаются отдельные биты в соответствии со значениями E_{max} и E_i, а затем эти массивы мажоритарно суммируются. В полученном эталонном массиве подсчитывается число ошибок — несовпадений с истинным массивом (при моделировании, в отличие от реальной ситуации, истинный массив доподлинно известен). Полученное значение интенсивности ошибок E_M сравниваются со значением E_i.

Если $E_i > E_M$ или $E_i < E_M$, то значение E_i уменьшается или увеличивается. Затем создается новый массив информации для i-го канала с меньшим или большим числом ошибок, вновь формируется мажоритарно-восстановленный массив, вычисляется новое значение E_M и т. д. Процесс заканчивается, когда E_i оказывается равным E_M с некоторой заданной точностью (точное равенство может быть никогда не получено). Порог вычисляется по формуле $Q = E_{max} - E_i$. Смысл заключительной операции состоит в том, что моделирующая программа «отодвигает» E_i от E_{max} до такой степени, когда мажоритарно-восстановленная информация становится не лучшей по качеству, чем информация в лучшем канале, а это и означает, что величина Q представляет собой искомый порог. По результатам моделирования создаются таблицы, по которым можно выбрать порог в разных ситуациях. Эти таблицы переносятся в память системы (см. рис. 6.48) и используются при выполнении программы рис. 6.49 для выбора порога при проверке условий ветвления.

6.31. Восстановление информации при ее приеме из параллельного канала связи [177]

При передаче информации через параллельный (например, байтовый) канал связи возможно появление ошибок — постоянных или кратковременных. Предлагается передавать каждый байт троекратно, сначала в прямом, затем в инверсном коде, а после этого вновь в прямом коде, но с переставленными разрядами. Перестановка разрядов осуществляется, например, циклическим сдвигом кода на один бит вправо. Приемник запоминает полученные байты, инвертирует второй и переставляет в обратном порядке разряды в третьем, чтобы получить три версии исходного байта. В отсутствие ошибок все они одинаковы, при наличии постоянной одиночной ошибки между ними появляются отличия. Эти три версии сопоставляются и «по большинству голосов» определяется истинное значение бита в каждом разряде.

Предположим, что передатчик посылает в канал связи исходный D, инвертированный $\bar{D}$ и циклически сдвинутый вправо на один разряд D* байты (рис. 6.51):

D =10010110,

$\bar{D}$ =01101001,

D*=01001011.

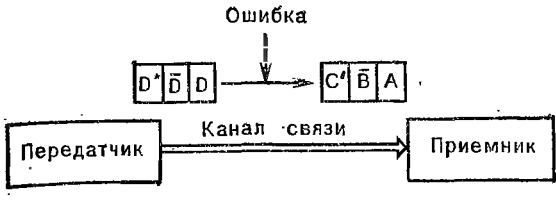


Рис. 6.51. Передача информации из передатчика в приемник.

Один и тот же байт пересылается по каналу связи три раза — в прямом, инверсном коде и после циклического сдвига разрядов, что позволяет восстановить ошибочный бит

При наличии постоянной ошибки типа лог. 1 в старшем (крайнем левом) разряде параллельного канала связи приемник получает следующую информацию (с искажениями, выделенными полужирным шрифтом):

A = 10010110 (ошибка совпала с истинным значением бита);

$\overline{B}$ = 11101001 (ошибка проявилась в «своем» разряде);

C′ = 11001011 (ошибка проявилась при передаче «чужого» разряда исходного кода).

После инвертирования кода $\overline{B}$ и циклического сдвига на один разряд влево кода C′ приемник получает для сопоставления следующие коды:

A = 10010110 — код без ошибок;

B = 00010110 — ошибка в старшем разряде;

C = 10010111 — ошибка в младшем разряде.

При мажоритарной обработке каждого разряда (по большинству голосов) ошибки исправляются.

Если ошибка не постоянная, а кратковременная, то она также устраняется. Предположим, что ошибка типа лог. 1 действует в пятом разряде (счет справа налево от нуля) на протяжении передачи первого и второго байтов, а третий байт передается без искажений. Приемник получает следующие коды:

A = 10110110,

$\overline{B}$ = 01101001,

C′ = 01001011.

После инвертирования кода $\overline{B}$ и циклического сдвига на один разряд влево кода C′ приемник получает два одинаковых правильных кода B = C = 10010110, что обеспечит правильное «большинство» при голосовании. Предлагаемый способ позволяет исправлять некоторые групповые ошибки.

6.32. Схема регистрации устойчивых одиночных ошибок в динамическом ОЗУ [184]

При работе динамического ОЗУ возможно появление ошибок в хранимой информации. Ошибки типа сбой устраняются повторной записью в ячейку исправленной информации, однако устойчивые ошибки, вызванные отказами элементов (транзисторов, токоведущих линий и т. п.), устранить таким путем невозможно. Опасность представляет накопление устойчивых ошибок, так как корректирующие возможности аппаратуры ограничены и начиная с некоторого момента она уже не сможет справиться с накопившимися отказами. Поэтому возникает задача аппаратной «сортировки» ошибок: одиночные сбои должны исправляться и отсеиваться как несущественные, а одиночные отказы — исправляться и регистрироваться с выдачей человеку-оператору соответствующей информации для ремонта:

Код Хэмминга позволяет исправлять одиночные и обнаруживать (но не исправлять) двойные ошибки. (Ошибки большей кратности могут не обнаруживаться.) При выявлении одиночной ошибки в считанном из ОЗУ слове она исправляется, слово передается в процессор и одновременно с этим возвращается в исходную ячейку памяти для устранения ошибки в самой ячейке. Таким образом, сбои ликвидируются по мере использования информации процессором.

Постоянная одиночная ошибка (отказ) характеризуется повторяемостью адреса и номера разряда ячейки памяти (если бы кто-то вел учет адресов и номеров ошибочных разрядов при работе микроЭВМ). В схеме рис. 6.52 при обнаружении одиночной ошибки (отказа или сбоя) срабатывает двухвходовый элемент И, в регистре RG запоминается адрес и номер ошибочного разряда ячейки ОЗУ. Если в дальнейшем ситуация повторяется, т. е. вновь обнаружена ошибка в том же разряде той же ячейки ОЗУ, то срабатывают компараторы, на выходе трехвходового элемента И формируется сигнал лог. 1 (CL — синхросигнал, гарантирующий окончание переходных процессов на выходах компараторов), триггер TT устанавливается в 0, информация об ошибке в регистре RG защищается от обновления, так как двухвходовый элемент И закрывается по нижнему входу.

Сигнал лог. 0 выхода триггера TT поступает в процессор и вызывает прерывание программы. Процессор считывает содержимое регистра RG и использует его для выдачи сообщения оператору. Для продолжения поиска постоянных ошибок процессор формирует сигнал установки триггера TT в 1.

Вероятность повторения случайных сбойных ситуаций (которые обычно связывают с пролетом α-частиц через кристалл микросхемы памяти) с повторяющимися адресом и номером ошибочного разряда практически равна нулю. Поэтому схему можно рассматривать как своеобразный фильтр для разделения одиночных ошибок типа сбой и отказ с регистрацией параметров последних.

Постоянная одиночная ошибка регистрируется триггером TT только в том случае, когда ошибочная ячейка считывалась дважды. Это не означает, однако, что считывалась одна и та же информация, так как между считываниями могла произойти и запись нового кода. Более того, возможно, что ячейка многократно использовалась для записи и считывания информации, но ошибка «пряталась», совпадая с истинным значением хранимого бита. Однако в конечном счете

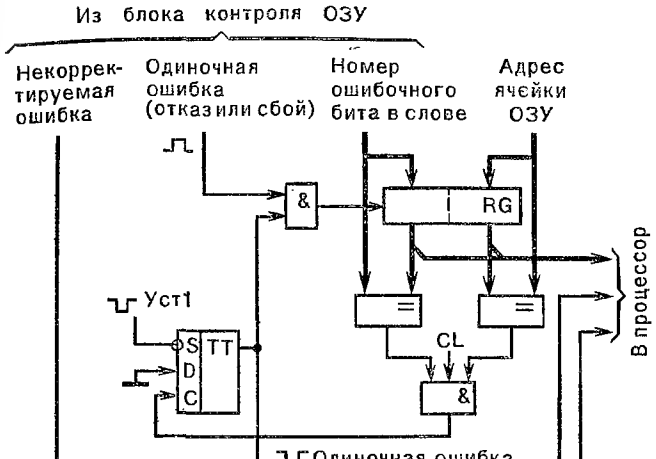


Рис. 6.52. Схема регистрации параметров одиночных ошибок.

Совпадение параметров двух последовательных ошибок расценивается как признак отказа

ошибка должна проявиться, когда хранимый бит «не совпадает» с ней. Таким образом, повторное проявление ошибки может отстоять от первого сколь угодно далеко; единственным условием срабатывания триггера является отсутствие иных одиночных ошибок в период между первым и вторым проявлением искомой. Если же таковые имеются, то схема будет терять предысторию и вновь пытаться ее накопить, запоминая параметры последней ошибки.

При обнаружении ошибки с кратностью, большей или равной двум, в процессор посылается сигнал прерывания с высоким уровнем приоритета, однако дополнительная информация об ошибке не формируется, чтобы не усложнять схему.

6.33. Схема слежения за адресами команд, выполняемых процессором [192]

При выполнении диагностических или отладочных процедур часто бывает необходимо регистрировать факт выборки или выполнения процессором команды с адресом, заданным человеком-оператором. В схеме рис. 6.53 адрес команды заносится в программно-доступный регистр RG. Если текущий адрес совпал с заданным, то срабатывает компаратор и в процессор посылается сигнал прерывания, по которому выполняется сервисная программа, предоставляющая оператору интересующую его информацию (содержимое регистров процессора, значения переменных и т. п.).

Схема рис. 6.53, однако, оказывается непригодной, если процессор обладает способностью выбирать команды «с опережением». Такой процессор одновременно с выполнением команды i дешифрирует ранее выбранную команду $i+1$ и выбирает из памяти команду $i+2$. Эти команды извлекаются из последующих соседних ячеек памяти в предположении, что участок программы линеен и в ближайшем будущем придется эти команды выполнять. Таким образом, при работе на линейном участке программы процессор наполняет конвейер команд, т. е. просматривает программу и готовит операнды, «заглядывая вперед». Однако если процессор уходит с линейного участка (например, встретив команду безусловной передачи управления, получив сигнал прерывания и т. п.), то накопленные в конвейере

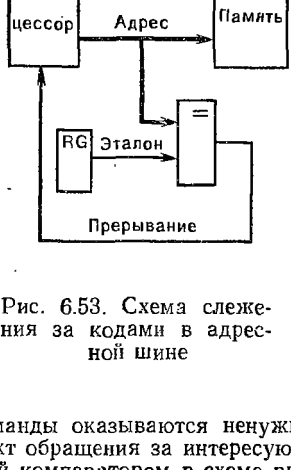


Рис. 6.53. Схема слежения за кодами в адресной шине

Рис. 6.54. Схема подмены кода команды по заданному адресу

команды оказываются ненужными и отвергаются. Иными словами, факт обращения за интересующей оператора командой (регистрируемый компаратором в схеме рис. 6.53) еще не означает, что команда будет принята к исполнению — возможно, что она будет отвергнута. В этом случае прерывание окажется ложным и пользователь будет введен в заблуждение.

Чтобы избежать регистрации ложных событий, можно было бы интересующую оператора команду заменить командой программного прерывания или какой-либо иной. Тогда, дойдя до такой команды и приняв ее к исполнению, процессор смог бы выполнить сервисные операции. Однако если программа записана в ПЗУ, то прямая подмена команд оказывается невозможной. В схеме рис. 6.54 осуществляется подмена кода команды, хранимой в памяти (ПЗУ или ОЗУ), кодом команды, хранимым в программно-доступном регистре RG2.

При нормальной работе сигнал несовпадения с выхода компаратора поддерживает мультиплексор MS в состоянии, при котором шины данных процессора и памяти соединены. При обнаружении совпадения мультиплексор вместо кода команды из памяти передает

в процессор код команды из регистра RG2. Если эта команда будет отвергнута процессором (из-за конвейерного принципа его работы), то ложной регистрации совпадения не произойдет. Если команда все же будет воспринята, то процессор, следуя ей, перейдет к выполнению сервисной программы.

Вместо команды программного прерывания можно, например, использовать команду ввода-вывода с адресом внешнего устройства (8 бит), на который реагирует дешифратор DC (рис. 6.54). При выполнении этой команды процессор через дешифратор вызывает прерывание работы по исследуемой программе и переход к сервисным операциям.

6.34. Байтовый канал связи с чередованием во времени проверок нечетности и четности числа передаваемых единиц [195]

Схема рис. 6.55 позволяет контролировать информацию при ее пересылке из передатчика в приемник и обнаруживать ошибки как в информационных, так и в управляющих цепях. Предполагается, что из передатчика в приемник пересылается массив байтов с условными номерами 0, 1, 2, ..., 2n—1 (последним передается байт с не-

Рис. 6.55. Схема передачи последовательности байтов с попеременным соблюдением и преднамеренным нарушением условия нечетности числа передаваемых единиц.

Обнаруживаются ошибки в цепях управления, нарушающие правильную последовательность байтов

четным номером, чтобы в следующем сеансе связи естественным образом начать с передачи байта с четным (нулевым) номером). Каждый байт, подготовленный к передаче и поступающий в линии D' или D'', сопровождается контрольным разрядом (K1 или K2), дополняющим общее число единиц в 9-разрядном коде до нечетного. В данном примере информация передается в прямом коде. Управляющие цепи контролируются благодаря чередованию во времени

байтов с четными и нечетными номерами (для краткости — четных и нечетных байтов), причем контрольные разряды тех и других передаются соответственно в прямом и инверсном коде, а приемник чередует проверки нечетности и четности числа принятых единиц.

В исходном состоянии триггер ТТ установлен в 0 ранее поступившим импульсным сигналом Сброс; M=0, двухвходовый элемент Исключающее ИЛИ (М2) подготовлен к передаче контрольного разряда без его инвертирования. Передатчик, сформировав массив информации, начинает передачу первого байта (с номером 0) по верхнему каналу (D′, К1). Для этого он открывает верхнюю группу передающих элементов сигналом V1=1, а затем через интервал времени, достаточный для окончания переходных процессов на верхнем входе элемента И—НЕ приемника, формирует отрицательный импульс С1. Фронт этого импульса вызывает прием информации в регистр RG и опрос схемы контроля (9-входового элемента Исключающее ИЛИ (М2) с инвертором на выходе). При правильной работе системы сигнал ошибки не формируется. По окончании импульса С1 триггер ТТ меняет состояние (M=1).

После передачи байта с номером 0 передается байт с номером 1 по нижнему каналу передатчика. Процесс протекает аналогично, однако контрольный разряд К2 дважды инвертируется — перед выдачей в канал связи и при приеме из канала (функция второго инвертора выполняется двухвходовым элементом М2, так как на его нижний вход подан сигнал M=1). Двойное инвертирование разряда К2 не меняет его значения, поэтому при правильной работе системы сигнал ошибки не формируется. По окончании импульса С2 триггер ТТ вновь меняет свое состояние (M=0) и приемник готов принять байт с номером 2, выдаваемый по верхнему каналу и сопровождаемый «правильным» значением контрольного разряда К1.

Таким образом, при передаче массива из чередующихся четных (0, 2, 4, ...) и нечетных (1, 3, 5, ...) байтов попеременно открываются верхняя и нижняя группы передающих элементов, а приемник приспосабливается к постоянно меняющемуся критерию ошибки (четность — нечетность числа единиц 9-разрядного кода в канале связи). Это позволяет помимо обнаружения одиночных информационных ошибок контролировать нарушение правильной последовательности байтов. Например, если пропущен (не выдан) нечетный байт, то в канале связи будут два раза подряд присутствовать четные байты, сопровождаемые «правильными» значениями контрольного разряда К1. В то же время приемник после получения «правильного» значения контрольного разряда ожидает «неправильное» (преднамеренно проинвертированное) и поэтому регистрирует ошибку из-за нарушения чередования четных и нечетных байтов.

Глава 7

Адресация, обмен данными

7.1. О преимуществах псевдослучайной адресации памяти

Для схемы, показанной на рис. 7.1, *а*, характерно упорядоченное подключение адресных и информационных выводов к соответствующим линиям адресной и информационной шин, а схема, показанная на рис. 7.1, *б*, разведена «хаотически». Предположим, что некий зло-

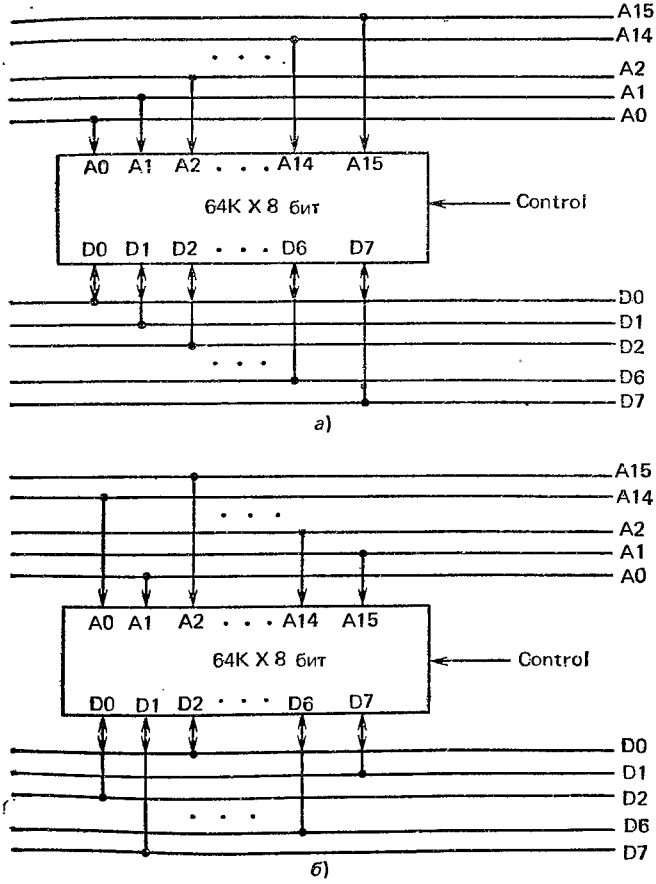


Рис. 7.1. Регулярная (*а*) и нерегулярная (*б*) разводка адресных и информационных выводов ОЗУ

умышленник в тайне от пользователя преобразовал (с помощью скальпеля и паяльника) первую схему во вторую в надежде на хаос в работе системы. Достиг ли он цели? Конечно, нет. Пользователь просто не заметит этих изменений. Каждому адресу, передаваемому по шине, соответствует единственный адрес, подаваемый на входы памяти. Биты слова данных при записи путаются и хранятся в «чужих» разрядах памяти, что не мешает, однако, автоматически восстановить исходное слово данных при его считывании.

Об этом свойстве универсальности выводов блоков памяти забывают даже опытные схемотехники — часто приходится слышать о «непреодолимых» трудностях при разводке печатных плат, при упорядочении шин для прокладки плоских кабелей и т. д.

Предположим теперь, что некоторый блок памяти используется как буфер: сначала в него записывается массив информации, а затем считывается в том же порядке, что и при записи (буфер типа FIFO). Обычно для указания последовательных адресов используется счетчик. Это своего рода даань здравому смыслу: мы просто привыкли к тому, что уровень воды в «буферной емкости» меняется плавно. При попытке ускорить работу буферной памяти мы в конце концов придем к проблеме ускорения переноса в счетчике, которая решается с использованием дополнительной аппаратуры [158].

Оказывается, что можно обойтись без дополнительной аппаратуры ускорения переноса в счетчике, если отказаться от самой идеи переноса. Вместо счетчика можно использовать для выработки цепочки адресов генератор псевдослучайных чисел — кольцевой сдвиговый регистр с элементами суммирования по модулю два в цепи обратной связи. После установки генератора в исходное состояние каждый управляющий импульс сдвига вызывает появление на выходах регистра очередного псевдослучайного числа [66]. При этом записываемая в буфер информация «разбрасывается» по псевдослучайным ячейкам памяти. При чтении массива генератор вновь устанавливается в исходное состояние и воспроизводит ранее записанную информацию в той же последовательности, что и при записи.

Использование псевдослучайной адресации видеопамяти растрового графического дисплея позволяет повышать живучесть системы отображения: выход из строя какого-либо банка памяти приведет к появлению на экране случайных мелких вкраплений, а не к исчезновению целого фрагмента изображения [67]. Кроме того, предложено хранить в видеопамяти либо негативное, либо позитивное изображение кадра в зависимости от преимущественного характера дефектов. Выбор того или иного режима производится после тестирования памяти с таким расчетом, чтобы помехи были представлены точками, совпадающими с фоном отображаемого объекта. При выдаче на экран первоначально проинвертированное (для «скрытия» дефектов) изображение вновь инвертируется, так что пользователь всегда работает с позитивом. При выборе режима также учитывается тот факт, что человеческий глаз лучше замечает черные точки на белом фоне, чем белые на черном.

7.2. Размещение оперативной памяти данных микроЭВМ вне ее адресного пространства [68]

Для работы с внешним ОЗУ объемом 16К 16-разрядных слов процессор микроЭВМ загружает начальный адрес в программно-доступный регистр RG (рис. 7.2), а затем устанавливает в 1 програм-

мно-доступный триггер ТТ, разрешая последующее прохождение сигнала прибавления единицы на счетный вход +1 регистра. Далее, выполняя команду типа регистр — память с обращением по некоторому адресу Ах, процессор мякроЭВМ фактически производит чтение или запись информации по адресу, который был указан в регистре RG. В конце обращения на счетный вход регистра поступает положительный фронт сигнала, который вызывает прибавление единицы к старому содержимому этого регистра. Поэтому последую-

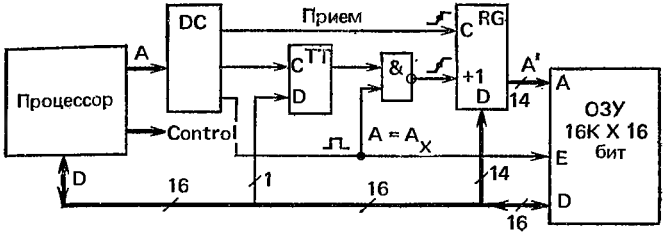


Рис. 7.2. Вариант подключения ОЗУ к процессору

щее обращение пропессора микроЭВМ по тому же адресу Ах вызовет выбор следующей ячейки из ОЗУ и т. д. Триггер ТТ может быть установлен программно в нулевое состояние, при котором запрещается операция прибавления единицы. Таким образом, 16К адресов внешнего ОЗУ практически не заннмают адресного пространства микроЭВМ. Точнее, заняты три адреса, соответствующие трем выходам дешифратора.

7.3. Расширение адресного пространства микроЭВМ при плотном его заполнении «окнами» [69]

В данном случае используется механизм расширения адресного пространства, по своей сути эквивалентный описанному ранее, в § 1.5. Схемная реализация также близка к рассмотренной ранее, поэтому здесь не приводится. На рис. 7.3 приведены диаграммы, поясняющие работу микроЭВМ с 16 блоками памяти, каждый из которых содержит 32К ячеек. Все адресное пространство мякроЭВМ (32К ячеек) разбито на 16 зон по 2К ячеек. Из каждой такой зоны можно провести «луч» и направить его в любую зону любого блока памяти. Одна из зон, например зона Y в адресном пространстве микроЭВМ, отводится для адресации регистров внешних устройств, в том числе регистров, определяющих направления лучей, возможность доступа к блокам памяти по чтению или (и) по записи и т. д. При включении напряжения питания управляющие регистры устанавливаются в определенные состояния, однозначно определяющие начальную конфигурацию системы. Дальнейшее управление «лучами» производится операционной системой.

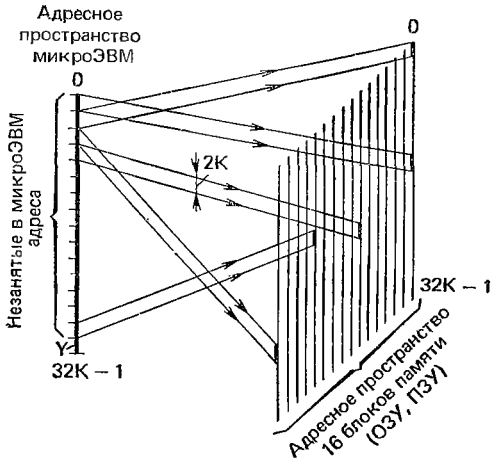


Рис. 7.3. Адресные пространства микроЭВМ и внешних памятей

7.4. Преобразователь логических адресов в физические, обеспечивающий конвейерный режим работы блоков памяти [70]

Задача состоит в частичном распараллеливания работы блоков памяти ОЗУ0—ОЗУ3. Мультиплексор М (рис. 7.4, *а*) выполняет преобразование адресов А, поступающих из центрального процессора (логических адресов), в физические адреса А′ таким образом, чтобы двум соседним логическим адресам соответствовали физические ячейки памяти, расположенные в разных блоках ОЗУ. Это позволяет частично распараллеливать работу этих блоков. Если, например, центральный процессор извлекает из памяти операнд, состоящий из четырех байтов, а каждое ОЗУ имеет однобайтовую структуру, то можно, послав в быстром темпе 4 последовательных логических адреса, запустить по очереди все 4 блока памяти. В течение некоторого времени они будут работать параллельно, а затем в быстром темпе по очереди (в порядке запуска) из блоков памяти в процессор по шине данных будут переданы все 4 байта. Этот механизм напоминает конвейер, с выхода которого в порядке поступления на вход выдаются нужные изделия.

На рис. 7.4, *б* показана схема преобразования логических адресов А в физические адреса А′. Из этой схемы видно, что, перебирая последовательно возрастающие логические адреса А, на выходах А′ мультиплексора получаем «расслсение» адресов на 4 группы, что и требовалось. (Эта процедура напоминает раздачу колоды карт четырем игрокам.) Возможны более сложные преобразования с использованием табличной памяти.

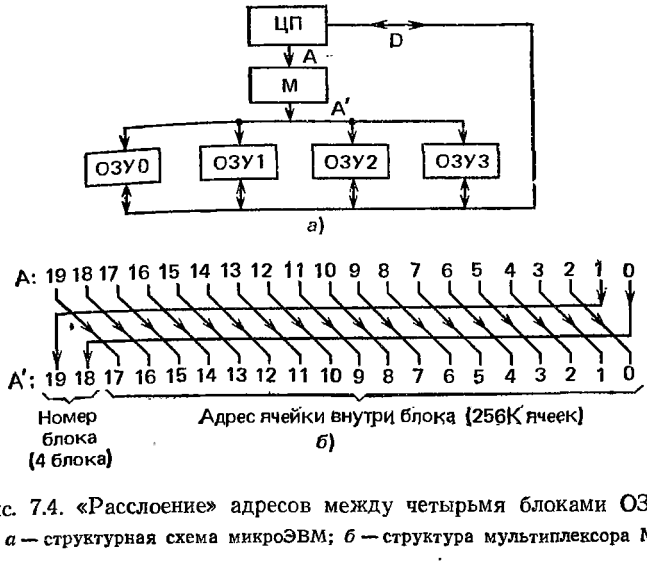


Рис. 7.4. «Расслоение» адресов между четырьмя блоками ОЗУ:
а — структурная схема микроЭВМ; *б* — структура мультиплексора М

7.5. Повышение производительности микроЭВМ за счет опережающего выбора команд и данных из памяти

Работа процессора по программе обычно связана с выбором информации (командной, числовой) из соседних ячеек памяти. Поэтому при обращении к i-й ячейке памяти существует высокая вероятность того, что данные из (i+1)-й ячейки пригодятся в ближайшем будущем.

Рассмотрим структуру памяти (ПЗУ или ОЗУ с преимущественным чтением), позволяющую в одном цикле обращения выбрать два слова: то, которое нужно «сейчас», и то, которое может пригодиться в «ближайшее время» [71]. В памяти (рис. 7.5) хранятся 1024 пары 16-разрядных слов. В каждой паре есть «основное» и «опережающее» слово. Так, при обращении к ячейке 1 в 32-разрядную шину данных будет выдана пара слов: «основное» слово B из ячейки 1 и слово C, которое «на самом деле» хранится в ячейке 2 и выбрано «на всякий случай». Поэтому если в дальнейшем потребуется слово C, то за ним уже не нужно будет обращаться в память, так как оно хранится в буферном регистре процессора. Если нужно продолжить последовательный выбор слов, то обращение будет идти по адресу 3 (а не 2) и при этом будут получены слова D и E и т. д. Таким образом, в благоприятных случаях число обращений может быть сокращено примерно вдвое по сравнению с числом обращений к обычной «однословной» памяти.

В отличие от ранее предлагавшихся схемных решений здесь не требуются схема быстрого прибавления единицы к адресу и схема мультиплексирования для правильного взаимного расположения выдаваемых слов.

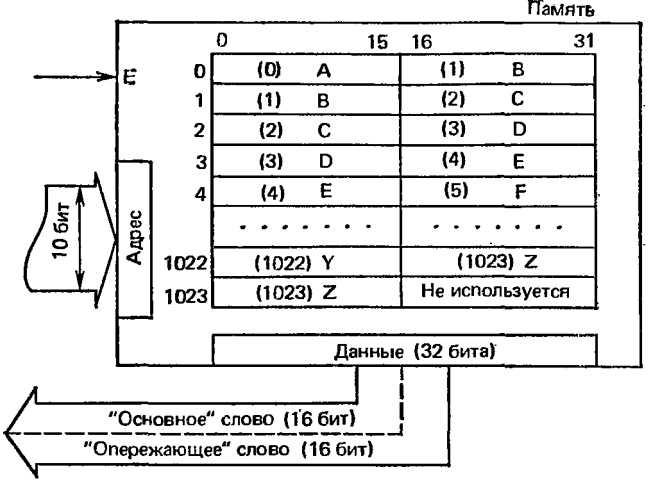


Рис. 7.5. Память с параллельным опережающим выбором информации

Другой вариант построения блока памяти с опережающим выбором даиных [127] показан на рис. 7.6.

При обращении процессора к блоку памяти на адресной шиие A магистрали устанавливается нужный код адреса, а затем формируется сигнал MSYN, фронт которого подтверждает истинность сигналов на адресной шиие (см. гл. 1,2). Старшие 4 разряда адреса распознаются дешифратором DC. Младшие 12 разрядов адреса сравниваются с кодом, который хранится в счетчике CT. Передающие элементы TR открываются сигналом F.

Если совпадения нет, то блок управления формирует импульсный сигнал L приема информации в счетчик CT. В результате адрес, выданный процессором в шину A, запоминается в счетчике и поступает на адресиые входы ПЗУ. Время считывания информации из ПЗУ (типа 2716), измеряемое от момента подачи адреса на входы до момента установления истинных сигналов на выходах, составляет 450 нс. Поэтому блок управления через интервал времени, равный 550 нс (100 нс — запас), формирует ответный сигнал SSYN, подтверждающий истинность данных, считанных из ячейки ПЗУ и переданных через элементы TR в шину данных D.

Процессор снимает сигнал MSYN с шины упяавления (Control) и адрес с шины A. Блок управления, в свою очередь, снимает сигнал F, закрывая передающие элементы TR, и формирует сигнал «+1», по которому к содержимому счетчика CT прибавляется единица. Одновременно с этим блок управления начинает отсчет времени (550 нс), подготавливаясь к выдаче в шину D кода из следующей ячейки ПЗУ, если это потребуется.

Если при следующем обращении к блоку памяти адрес, выданный процессором в шину A, совпал с адресом, который хранится

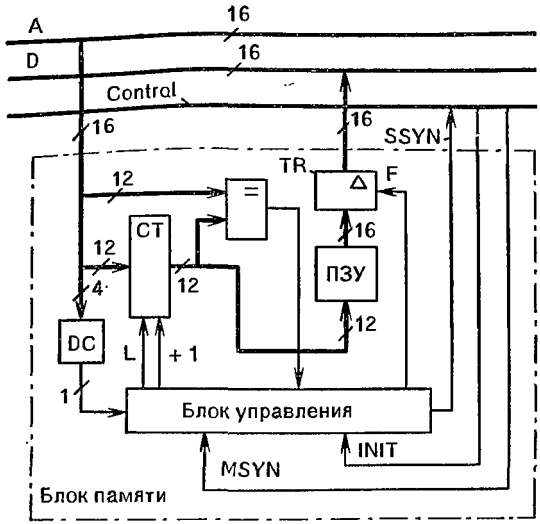


Рис. 7.6. Память с последовательным опережающим выбором информации

в счетчике СТ, то это означает, что затребована информация из соседней ячейки. При опознании адреса блока памяти формируется сигнал F и выходы ПЗУ подключаются к шине D.

Как было показано, информация из нужной ячейки либо уже считана, если истекли 550 нс с момента установления нового адреса на входах ПЗУ, либо будет считана в ближайшее время, когда период 550 нс будет исчерпан. Если информация готова, то с некоторой задержкой (около 100 нс) формируется сигнал SSYN, подтверждающий ее истинность. Если информация еще не готова, то блок управления продолжает отсчет времени и по истечении 550 нс формирует сигнал SSYN. При завершении обмена к содержимому счетчика СТ вновь прибавляется единица, вновь начинается отсчет времени (550 нс) и т. д. Таким образом, блок памяти постоянно стремится «опередить события», подготавливая очередное слово данных в расчете на то, что именно оно будет затребовано.

Эффективность использования предложенного метода зависит от особенностей построения конкретной микроЭВМ. Согласно [127], при использовании блока памяти для хранения программ время выполнения программы уменьшается на 1—18 % в зависимости от степени регулярности ряда адресов обращения к ПЗУ. Выигрыш 18 % получен при выполнении программ, в которых 95 % адресов образуют последовательную цепочку; выигрыш 8—10 % достигается при выполнении программ, в которых только 50 % адресов образуют последовательную цепочку.

Если при выполнении программы используется ряд констант, образующих непрерывный массив ячеек, то этот массив целесообраз-

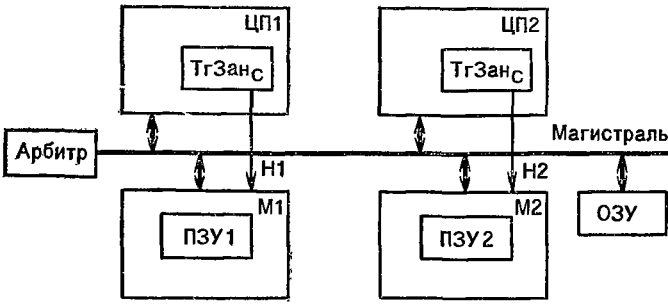


Рис. 7.7. Двухпроцессорная система с индивидуальными ПЗУ и общим ОЗУ

но хранить в отдельном блоке памяти рассмотренного типа. Такое разделение программ и констант позволит уменьшить нерегулярность последовательности адресов в каждом блоке и, следовательно, повысить производительность мякроЭВМ.

На рис. 7.7 показана двухпроцессорная система, в которой процессор ЦП1 — главный, а процессор ЦП2 выполняет функции контроллера дисплея. Блоки памяти М1 и М2 выполнены по схеме, приведенной на рис. 7.6. Программы, выполняемые процессорами ЦП1 и ЦП2, хранятся соответственно в ПЗУ1 и ПЗУ2. Обмен информацией между процессорами ЦП1 и ЦП2 осуществляется через общее ОЗУ. Захват магистрали процессором ЦП1 и ЦП2 производится с разрешения арбитра (см. гл. 2).

В отличие от схемы, приведенной на рис. 7.6, блоки памяти М1 и М2 содержат дополнительный вход, на который подается сигнал занятости магистрали с триггера $ТгЗан_С$ схемы захвата магистрали соответствующего процессора (пример построения схемы захвата магистрали был приведен на рис. 2.23). Если магистраль занята процессором ЦП1, то Н1=1, Н2=0, активизирован блок М1. Если магистраль занята процессором ЦП2, то Н2=1, Н1=0, активизирован блок М2.

Благодаря разделению программ обоих процессоров и их размещению в разных блоках памяти (М1 и М2), в каждом из этих блоков обеспечивается ускоренный доступ к информации за счет опережающего выбора очередного слова из ПЗУ. Кроме того, за счет взаимоисключающей активизации блоков М1 и М2 эти блоки могут занимать одну и ту же область адресного пространства микроЭВМ, например, область 8000—BFFF. Это позволяет упростить систему адресации и не вводить в адресную шину дополнительные линии.

7.6. Гибкое использование абсолютных адресов, хранимых в программном ПЗУ [72]

Обычно в системе команд процессора присутствуют команды типа регистр — память, в которых абсолютный адрес записан непосредственно в команде:

КОП	R_i
Абсолютный адрес	

Команда в соответствии с кодом операции КОП может быть такой: извлечь содержимое ячейки памяти по указанному в команде абсолютному адресу, сложить полученное число с содержимым внутреннего регистра R_i процессора и результат поместить в этот же регистр.

Если программа, в которой используются команды такого типа, хранится в ПЗУ, то абсолютные адреса, указанные в этих командах, не подлежат изменению. Это приводит к «жесткости» структуры данных, ее точной привязке к определенным адресам памяти. Предлагаемое решение (рис. 7.8) позволяет работать с перемещаемыми

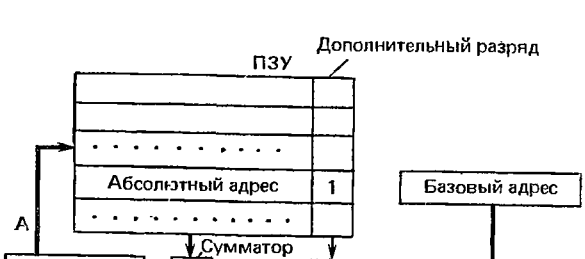


Рис. 7.8. Схема модификации адресов

структурами данных. Абсолютные адреса, указанные в соответствующих командах, помечаются единицами в дополнительно введенном разряде ПЗУ. Процессор загружает в программно-доступный регистр некоторый базовый адрес, например код 500_{10}. В дальнейшем при обнаружении единицы в дополнительном разряде ПЗУ замыкается ключ (группа логических элементов) и к абсолютному адресу с помощью сумматора прибавляется базовый адрес. Если, например, абсолютный адрес, хранимый в ПЗУ, равен 850_{10}, то результирующий адрес, считанный процессором, будет равен 1350_{10}.

Таким образом, процессор, желая обрабатывать данные, смещенные в памяти на величину Δ относительно той позиции, которая предполагалась при кодировании ПЗУ, должен просто записать в регистр значение Δ. При $\Delta=0$ абсолютные адреса, закодированные в ПЗУ, передаются в процессор без изменения.

7.7. Улучшенная структура буфера типа FIFO [128]

Рассмотрим сначала традиционную схему буфера, приведенную на рис. 7.9, *а*. Улучшения, которые будут рассмотрены позже, показаны штриховыми линиями.

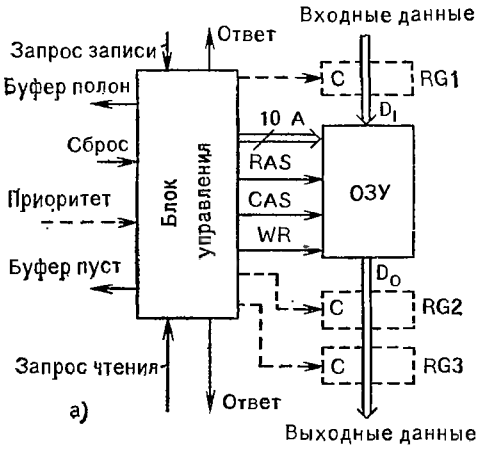


Рис. 7.9. Структура буфера типа FIFO *(а)* и диаграмма *(б)*, поясняющая его работу («гонка за лидером»).

Доработки, показанные штриховыми линиями, обеспечивают одновременный доступ к буферу со стороны источника и приемника информации, а также уменьшают среднее время доступа

Буфер типа FIFO («первый вошедший первым выходит») предназначен для временного хранения данных при их передаче от одного устройства к другому. Входные данные накапливаются в буфере с темпом, зависящим от особенностей устройства — источника информации (и, конечно, от временных характеристик самого буфера). Устройство-приемник извлекает данные из буфера в порядке их по-

ступления. Темп считывания данных зависит от временных характеристик устройства-приемника. В правильно спроектированной системе обмена данными объем буфера выбирается достаточным для того, чтобы он не переполнялся, иначе либо часть входных данных будет потеряна, либо источник информации будет работать с замедлением из-за ожидания моментов частичного освобождения буфера. (Фактически работа системы в этом случае будет такой же, как и в отсутствие буфера.)

Схема, показанная на рис. 7.9, *а*, содержит ОЗУ и блок управления. В ОЗУ хранятся передаваемые данные; блок управления получает запросы на запись (со стороны источника) и чтение (со стороны приемника) очередного слова данных, обслуживает эти запросы и посылает ответные сигналы источнику и приемнику информации. Блок управления формирует также сигналы Буфер полон и Буфер пуст, которые могут быть использованы источником и приемником информации (см. § 7.35).

Работа буфера поясняется диаграммой, показанной на рис. 7.9, *б*, где ОЗУ представлено кольцом из ячеек, имеющих адреса 0, 1, 2,..., 2^n-1, (n — разрядность адреса). При n=20 число ячеек составляет $2^{20}=1M$; разрядность данных в ячейке может быть равной 16.

Блок управления содержит два адресных регистра — указатель адреса чтения и указатель адреса записи. Указатель адреса чтения хранит адрес ячейки ОЗУ, из которой предстоит извлечь очередное слово данных по запросу устройства-приемника информации. Указатель адреса записи хранит адрес ячейки ОЗУ, в которую предстоит записать очередное слово данных по запросу устройства-источника информации. Содержимое указателей адресов чтения и записи увеличивается на единицу после выполнения соответствующих операций. В процессе работы устройства оба указателя перемещаются по диаграмме (рис. 7.9, *б*) по часовой стрелке, возможно, с разным темпом. При этом указатель адреса чтения «пытается догнать» указатель адреса записи, который, в свою очередь, от него «убегает». Такие гонки могут длиться неограниченно долго, причем оба участника периодически пробегают по всем адресам ОЗУ. Чем больше разрыв дистанции между участниками, тем больше данных хранится в буфере; на рис. 7.9, *б* указателю адреса записи удалось «оторваться» от указателя адреса чтения на 775—3=772 ячейки. Это означает, что в текущий момент в буфере хранится 772 16-разрядных слова.

Если указателю адреса чтения удалось догнать указатель адреса записи, то это означает, что буфер пуст. Если же указатель адреса записи настолько оторвался от указателя адреса чтения, что «обошел его на круг», то это означает, что все 2^n ячек буфера заполнены полезными данными. Обе ситуации обнаруживаются блоком управления по совпадению кодов в обоих указателях. Если совпадению предшествовала операция чтения, то буфер пуст и обслуживание очередного запроса чтения откладывается до тех пор, пока в буфере не появится хотя бы одно слово данных от источника информации. Если совпадению предшествовала операция записи, то буфер полон и новый запрос записи не обслуживается до тех пор, пока не будет выполнена хотя бы одна операция чтения со стороны приемника информации.

В исходном состоянии после воздействия сигнала Сброс указатели адресов чтения и записи содержат одинаковые, например, нулевые коды, сформирован признак Буфер пуст, запрос чтения не об-

служивается, пока буфер не пополнится хотя бы одним словом данных.

Блок управления выполняет функции арбитра по отношению к запросам записи и чтения (для устранения конфликтов между этими запросами), а также фуикции регенерации ОЗУ, если оно, как здесь предполагается, выполнено на микросхемах динамической памяти. Сигналы мультиплексированного адреса A (две последовательные посылки по 10 разрядов), выбора строки RAS, выбора столбца CAS и записи WR, показанные на рис. 7.9, *а*, вырабатываются блоком управления и передаются на соответствующие входы микросхем динамической памяти (более подробно об этих сигналах см. § 7.24). Отметим, что вместо регистров-счетчиков в качестве указателей адресов чтения и записи могут использоваться два одинаковых генератора псевдослучайных чисел (см. § 7.1).

Рассмотренная схема буфера имеет следующие недостатки. Во-первых, время обслуживания запроса чтения информации относительно велико. В худшем случае время обслуживания включает время выполнения текущего цикла регенерации очередной строки накопительных матриц микросхем памяти, время выполнения записи входных данных в буфер (если запрос записи поступил чуть раньше запроса чтения или приоритетная схема отдала предпочтение запросу записи) и время выборки данных из ОЗУ. Таким образом, время обслуживания запроса чтения может составить, например, 1 мкс. Во-вторых, невозможна одновременная обработка запросов записи и чтения.

Эти недостатки устраняются введением в схему дополнительных регистров, в данном примере — одного входного (RG1) и двух последовательно включенных выходных регистров (RG2 и RG3) с параллельным приемом информации по сигналу на входе C.

Логика работы блока управления несколько усложняется и обеспечивает выполнение следующих действий.

При поступлении запроса записи проверяется возможность приема входных данных в регистр RG1. Если содержимое регистра RG1 уже было переписано в ОЗУ, то этот регистр считается незанятым. В этом случае входные данные (одно слово) записываются в регистр RG1 и к источнику информации посылается ответный сигнал, подтверждающий выполнение операции. В дальнейшем блок управления при первой возможности освобождает регистр RG1, т. е. переписывает его содержимое в ОЗУ по адресу, который хранится в регистре-указателе адреса записи.

Регистры RG2 и RG3 хранят два начальных слова из блока данных, размещенного в ОЗУ. В примере, приведенном на рис. 7.9, *б*, содержимое ячейки с адресом 3 находится в регистре RG3, а содержимое ячейки с адресом 4 — в регистре RG2. Указатель адреса чтения соответственно перемещен на две позиции по часовой стрелке и подготовлен к выбору ячейки с адресом 5.

При поступлении запроса чтения блок управления анализирует свое состояние, проверяя готовность регистра RG3 к выдаче информации. Если в регистре RG3 имеется начальное слово блока данных, то оно сразу же выдается потребителю вместе с ответным сигналом. После снятия запроса чтения содержимое регистра RG2 переписывается в регистр RG3, а в регистр RG2 при первой возможности считывается информация из соответствующей ячейки ОЗУ, в нашем примере — из ячейки с адресом 5. Буфер вновь подготовлен к быстрой реакции на запрос чтения.

Введенные в схему регистры RG1—RG3 представляют собой, по существу, «буфер второго порядка» (буфер буфера) и позволяют использовать «свободное время» устройства для выполнения полезной работы — заблаговременного считывания данных, которые будут затребованы в ближайшем будущем, а также записи ранее полученного слова из входного регистра в ячейку ОЗУ.

Вход Приоритет определяет порядок работы блока управления с ОЗУ при возникновении конфликтных ситуаций. В зависимости от значения сигнала на этом входе предпочтение отдается операции записи или считывания информации. Воздействуя на вход Приоритет, можно до некоторой степени влиять на уровень заполнения буфера, создавая более благоприятные условия процессу пополнения или рассасывания массива хранимых данных. Например, если буфер заполнен слабо, то имеет смысл «поощрять» его пополнение. С другой стороны, если буфер близок к переполнению, то в конфликтных ситуациях следует отдавать предпочтение его рассасыванию. В простейшем случае буфер делится на две равные части, и в зависимости от того, превышен ли половинный уровень, выбирается значение управляющего сигнала Приоритет. Возможны, конечно, и другие способы получения этого сигнала. Однако в любом случае для определения степени заполнения буфера придется использовать дополнительные схемы, анализирующие разность чисел, хранимых в регистрах-указателях адресов записи и чтения, причем следует учитывать, что эти числа «вращаются по кругу» и периодически проходят через нулевую точку (см. рис. 7.9, *б*).

7.8. Оперативная память с возможностью одновременного чтения нескольких ячеек [74]

Рассматривается память с переменной структурой, при записи — последовательной (рис. 7.10, *а*), при чтении — параллельной (рис. 7.10, *б*). При записи блоки A, B и C расположены в адресном пространстве последовательно. При чтении эти же блоки (каждый емкостью 160К×16 бит) расположены параллельно и занимают адресное пространство емкостью 160К ячеек. Формат выдаваемого при чтении слова — 48 бит, по 16 бит из каждого блока. При записи, например, сначала загружается ячейка X, затем — ячейка X′ и затем ячейка X″. При чтении по адресу X выбираются все три загруженные ранее ячейки одновременно.

Такой механизм реконфигурации оказывается полезным в тех случаях, когда есть резерв времени для заполнения памяти, но его нет при выдаче информации. Например, блоки A, B и C могут хранить три цветовые составляющие цветного изображения. Выдача информации должна производиться синхронно из всех трех блоков в быстром темпе, чтобы ее поступление успевало за тремя синхронно бегущими лучами электронно-лучевой трубки. Запись информации может производиться в сравнительно медленном темпе, например, в периоды обратного хода строчной и кадровой развертки.

Схемная реализация адресного дешифратора показана на рис. 7.10, *в*. Дешифрации, как обычно, подлежат старшие разряды адреса, однако в данном случае имеются два дешифратора. Выбор сигнала с того или другого определяется состоянием управляющей линии Чтение — запись (Чт/Зп).

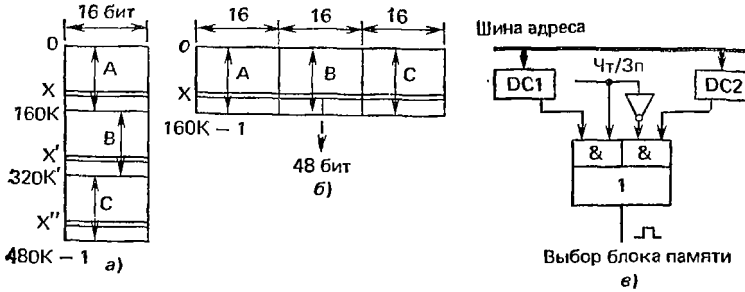


Рис. 7.10. Память с переменной структурой

7.9. Определение истинности передаваемой по магистрали информации без использования сопровождающего синхросигнала [78]

В отличие от общепринятого способа передачи информации по магистрали (см. гл. 1, 2), при котором ее истинность подтверждается специальным магистральным синхросигналом, предложен другой способ, при котором приемник сам определяет момент готовности информации (рис. 7.11). В исходном состоянии в магистрали типа «открытый коллектор» поддерживаются напряжении низкого уровня (сигналы лог. 0) за счет инвертирования единичного сигнала, по-

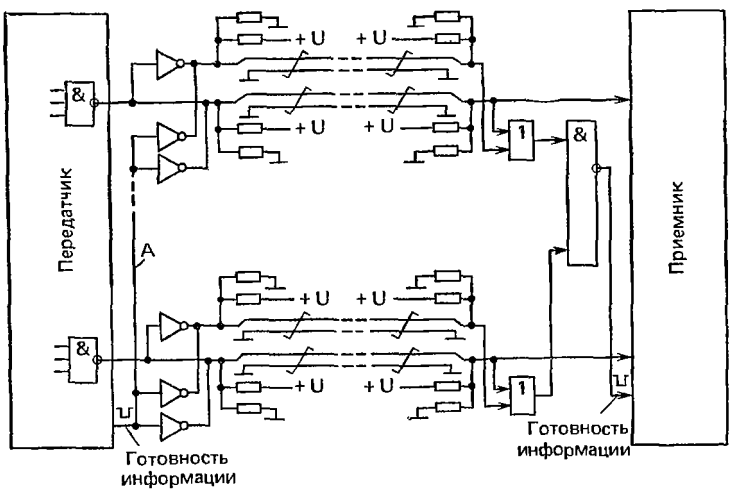


Рис. 7.11. Система с автоматическим определением момента готовности информации

данного в точку А (отсутствие готовности информации). На выходах элементов ИЛИ сформированы лог. 0, на выходе элемента И—НЕ — лог. 1 (отсутствие готовности информации).

Сеанс связи начинается с того, что передатчик пытается выставить в магистрали нужную информацию, хотя она «подавлена» сигналами лог. 0 с выходов инверторов, входы которых объединены в точке А. После начала такой попытки выдачи информации в магистраль передатчик выдает в точку А лог. 0 — сигнал готовности информации. Магистраль освобождается от «давления» инверторов, и в каждой битовой паре линий вместо кода 00 устанавливается код 01 либо 10. Появление любой указанной комбинации в некоторой паре линий вызывает формирование лог. 1 на выходе соответствующего элемента ИЛИ. После установления информации на всех без исключения линиях магистрали на входах элемента И—НЕ собирается группа лог. 1 и, следовательно, на его выходе формируется сигнал готовности информации (лог. 0).

Использование такого способа передачи информации позволяет работать с разноскоростными устройствами без потерь времени, имеющих место при расчете на «худший случай». Более того, можно работать с устройствами, временные характеристики которых заранее неизвестны либо меняются в широких пределах (например, при модернизации системы).

7.10. Автоматическое назначение собственных адресов блоков памяти микроЭВМ при включении напряжения питания [129]

Память микроЭВМ может состоять из отдельных функционально законченных блоков унифицированной конструкции. Пользователь по мере необходимости набирает из таких блоков память нужного объема. Блоки памяти могут быть одинакового или разного объема.

Для задания собственного адреса блока обычно используются микропереключатели или проводные перемычки, которые должны быть установлены в определенные положения. Это создает неудобства в работе, так как при изменении конфигурации микроЭВМ необходимо вычислять граничные адреса блоков памяти (здесь возможны ошибки вычислений), изучать техническую документацию на микроЭВМ и тщательно контролировать выполняемые операции «перекоммутации» (здесь, несмотря на контроль, также возможны ошибки).

Предлагается проводить автоматическое назначение собственных адресов блоков памяти, при котором человек-оператор не участвует в этой работе. Такое назначение производится с помощью специальной схемы, которая вступает в фазу активности только при включении напряжения питания микроЭВМ, а затем отключается. На рис. 7.12 показана часть микроЭВМ, имеющая непосредственное отношение к операции назначения собственных адресов блоков памяти. В данном примере микроЭВМ содержит два блока памяти. В блоке 1 размещены три накопителя ОЗУ1—ОЗУ3, каждый объемом 256К 16-разрядных слов. Блок 2 выполнен на основе пяти накопителей ОЗУ4—ОЗУ8, каждый объемом 64К 16-разрядных слов. МикроЭВМ может наращиваться аналогичными или иными блоками, в которых объем накопителя кратен 64К 16-разрядных слов.

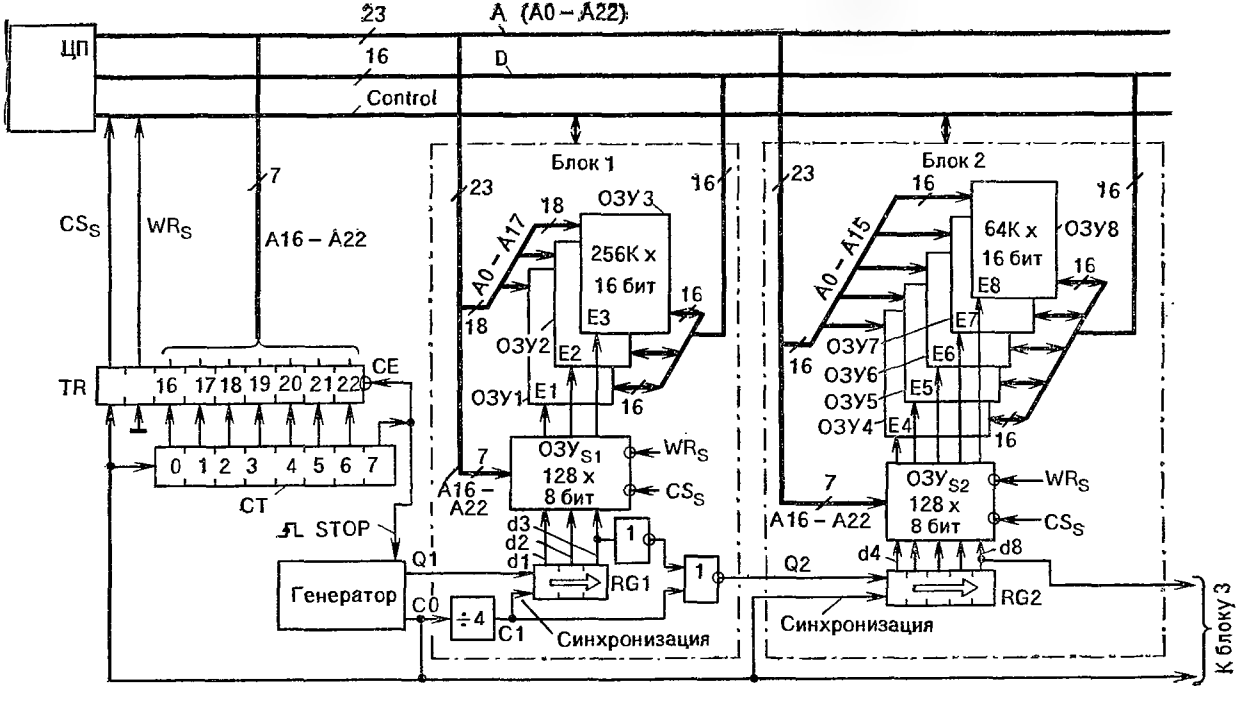


Рис. 7.12. МикроЭВМ с автоматическим назначением собственных адресов блоков памяти.
В качестве адресных дешифраторов использованы статические ОЗУ (ОЗУ$_{S1}$, ОЗУ$_{S2}$)

Адресная шина магистрали содержит 23 линии. Объем адресного пространства микроЭВМ составляет 2^{23} ячеек. Адресное пространство может быть представлено в виде 2^7 страниц, по 2^{16}=64К ячеек на странице. Номер страницы определяется старшими разрядами А16—А22 адресной шины; разряды А0—А15 определяют адрес ячейки на странице.

Каждый из трех накопителей в блоке 1 соответствует четырем страницам; блок 1 в целом занимает 3×4=12 страниц по 64К ячеек. Номера страниц, соответствующих блоку 1, следующие: 0000000, 0000001, 0000010,..., 0001011. Каждый из пяти накопителей, размещенных в блоке 2, соответствует одной странице адресного пространства. Номера страниц, соответствующих блоку 2, следующие: 0001100, 0001101, 0001110, 0001111, 0010000. Приведенное распределение адресного пространства для наглядности отражено в табл. 7.1.

Продвигаясь по адресному пространству с шагом 64К ячеек, сначала попадаем в область, занимаемую ОЗУ1 (4 шага), затем —

Таблица 7.1

Номер страницы адресного пространства (код в разрядах А22, А21,..., А16)							Сигналы выбора накопителя								Выбираемый накопитель
							E1	E2	E3	E4	E5	E6	E7	E8	
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	ОЗУ1
0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	
0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	
0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	
0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	ОЗУ2
0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	
0	0	0	0	1	1	0	0	1	0	0	0	0	0	0	
0	0	0	0	1	1	1	0	1	0	0	0	0	0	0	
0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	ОЗУ3
0	0	0	1	0	0	1	0	0	1	0	0	0	0	0	
0	0	0	1	0	1	0	0	0	1	0	0	0	0	0	
0	0	0	1	0	1	1	0	0	1	0	0	0	0	0	*
0	0	0	1	1	0	0	0	0	0	1	0	0	0	0	ОЗУ4
0	0	0	1	1	0	1	0	0	0	0	1	0	0	0	ОЗУ5
0	0	0	1	1	1	0	0	0	0	0	0	1	0	0	ОЗУ6
0	0	0	1	1	1	1	0	0	0	0	0	0	1	0	ОЗУ7
0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	ОЗУ8
0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	—
0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	—
1	1	⋮	⋮	⋮	1	1	0	0	0	0	0	0	0	0	—
							$ОЗУ_{S1}$			$ОЗУ_{S2}$					

ОЗУ2 и т. д. Таблица, таким образом, однозначно определяет функции аппаратуры дешифрации адресов. С подобной таблицей нам уже приходилось работать при использовании ПЗУ в качестве адресного дешифратора (см. § 1.4). В данном случае предлагается вместо ПЗУ использовать статические ОЗУ ($ОЗУ_{S1}$ и $ОЗУ_{S2}$), в которых после загрузки (по включению напряжения питания) постоянно хранится информация в соответствии с табл. 7.1: в $ОЗУ_{S1}$ хранятся «столбцы» Е1—Е3, а в $ОЗУ_{S2}$ — «столбцы» Е4—Е8 таблицы.

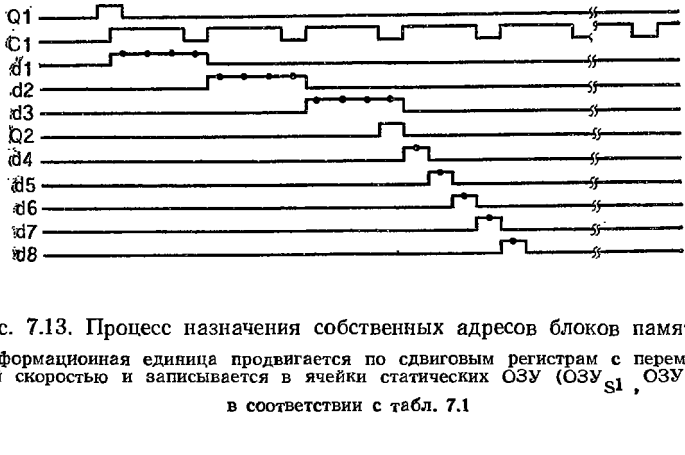


Рис. 7.13. Процесс назначения собственных адресов блоков памяти.

Информационная единица продвигается по сдвиговым регистрам с переменной скоростью и записывается в ячейки статических ОЗУ ($ОЗУ_{S1}$, $ОЗУ_{S2}$ в соответствии с табл. 7.1

Дешифрация осуществляется следующим образом. Предположим, что из центрального процессора ЦП в адресную шину выдан некоторый адрес, содержащий в старших разрядах код 0001011. В ответ на этот код из $ОЗУ_{S1}$ и $ОЗУ_{S2}$ выбирается информация, показанная в строке табл. 7.1, помеченной знаком *. Из $ОЗУ_{S1}$ выбирается код Е1 Е2 Е3=0 0 1, из $ОЗУ_{S2}$ выбирается нулевой код. В результате сигналом Е3=1 активизируется накопитель ОЗУ3, из которого в соответствии с 18 младшими разрядами адреса выбирается нужная ячейка для записи или чтения ее содержимого.

Отметим, что для выбора накопителя ОЗУ3 (ОЗУ2, ОЗУ1) объемом 256К ячеек было бы достаточно анализировать с помощью $ОЗУ_{S1}$ не 7, а 5 разрядов адреса (А18—А22); однако для унификации аппаратуры микросхемы $ОЗУ_{S1}$ и $ОЗУ_{S2}$ выбраны однотипными. После загрузки информации в соответствии с табл. 7.1 в микросхемы $ОЗУ_{S1}$ и $ОЗУ_{S2}$ эти микросхемы работают только в режиме считывания независимо от типа операции (чтение или запись), выполняемой процессором.

Начальная загрузка микросхем $ОЗУ_{S1}$ и $ОЗУ_{S2}$ адресной информацией производится следующим образом (рис. 7.13).

После включения напряжения питания микроЭВМ центральный процессор в течение некоторого времени (например, в течение первой секунды) не проявляет активности, счетчик СТ установлен в состояние 11111111 (255_{10}), сдвиговые регистры RG1, RG2 установлены в 0 (цепи начальной установки не показаны).

Процесс загрузки начинается с пуска генератора импульсов, например через 0,1 с после установления номинальных питающих напряжений. Генератор вырабатывает сигналы C0 и Q1 (см. рис. 7.12). Положительные фронты сигналов C0 вызывают увеличение содержимого счетчика СТ на единицу. Первый положительный фронт сигнала C0 переводит счетчик СТ из состояния 255 в состояние 0. (Эти и последующие состояния счетчика показаны в верхней части рис. 7.13.) Делитель частоты (÷4) преобразует сигнал C0 в сигнал C1, так что синхронизация сдвигового регистра RG1 осуществляется в 4 раза медленнее, чем синхронизация регистра RG2. (Это связано с 4-кратным различием объемов накопителей в блоках 1 и 2: 256K и 64K слов.)

После перехода счетчика СТ в состояние «0» по входу CE открывается блок магистральных передатчиков TR; в линию записи информации в статические ОЗУ ($ОЗУ_{S1}$, $ОЗУ_{S2}$) поступает сигнал WR_S подготавливающий микросхемы $ОЗУ_{S1}$ и $ОЗУ_{S2}$ к выполнению операций записи. Импульсы C0, проходя через блок TR, воздействуют на входы выбора кристалла CS_S микросхем $ОЗУ_{S1}$ и $ОЗУ_{S2}$ и побуждают их выполнять операцию записи кодов с выходов d_i сдвиговых регистров RG1 и RG2 по адресам, вырабатываемым счетчиком СТ. Каждый отрицательный фронт сигнала CS_S подтверждает истинность адресного кода, выдаваемого со счетчика СТ через блок TR в линии A16—A22 старших разрядов адресной шины.

Сигнал Q1 в «медленном» темпе проходит через сдвиговый регистр RG1 и затем через элемент ИЛИ—НЕ поступает на информационный вход сдвигового регистра RG2. Далее информационный сигнал в более быстром темпе проходит через сдвиговой регистр RG2 и поступает в следующий блок, если он подключен к микроЭВМ.

Моменты записи информации с входов d_i в $ОЗУ_{S1}$ и $ОЗУ_{S2}$ показаны на рис. 7.13 точками. Сравнив временные диаграммы с информацией, представленной в табл. 7.1, можно отметить их тесную смысловую взаимосвязь. Так, например, строка, помеченная в табл. 7.1 знаком * , соответствует состоянию счетчика СТ, равному 11_{10} (см. рис. 7.13). При этом состоянии счетчика в ячейки с адресом $11_{10} = 1011_2$ микросхем $ОЗУ_{S1}$ и $ОЗУ_{S2}$ одновременно записывается информация d1 d2 d3=0 0 1 и d4 d5...d8=0 0...0, что и требуется в соответствии с данной строкой таблицы.

После кода 127_{10} счетчик СТ формирует код 128_{10} (единица в разряде 7 и нули в остальных разрядах), в результате блок TR «наисегда» закрывается, генератор импульсов по фронту сигнала STOP прекращает работу, т. е. схема начальной загрузки адресной информации в статические ОЗУ отключается от магистрали, выполнив свою задачу.

Таким образом, пользователь может не заботиться о состоянии адресных переключателей (так как их просто нет) и компоновать память микроЭВМ из унифицированных блоков (возможно, с разными объемами накопителей), располагая их вплотную, в произвольном порядке.

7.11. Помехоустойчивая магистраль типа Монтажное ИЛИ с использованием дифференциальных пар [81]

На рис. 7.14 представлена схема одной линии магистрали, объединяющей различные устройства микроЭВМ. В пассивном состоянии транзисторы VT1 и VT2 передатчика выключены, падение напряжения на резисторе R2 достаточно для поддержания на выходе диффе-

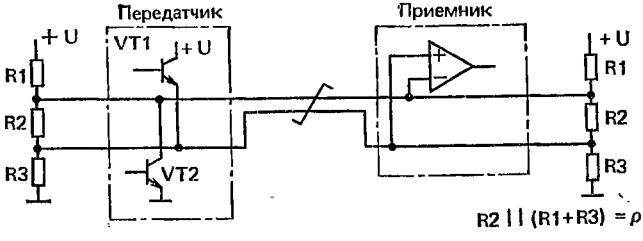


Рис. 7.14. Схема линии магистрали

ренциального усилителя приемника лог. 0 (напряжение низкого уровня). При срабатывании передатчика одновременно открываются транзисторы VT1 и VT2, при этом знак напряжения на входах приемника меняется. Обладая всеми достоинствами магистралей типа «дифференциальная пара» (см. § 4.4), эта магистраль позволяет подключать по схеме Монтажное ИЛИ до 24 передатчиков при работе на частотах свыше 10 МГц.

7.12. Арбитр распределенного типа [82]

Функции арбитра были рассмотрены в гл. 2. На рис. 7.15 показан фрагмент схемы распределенного арбитра, относящийся к одному из конкурирующих устройств; резисторы R1 подключены к физическим концам магистральной линии BR.

В исходном состоянии сигнал $P^{B}{}_{X}=1$, R=1 (запрос от «нашего» устройства отсутствует), BR=1 (магистраль не занята), a=0, $P_{ВЫХ}=1$, G=1 (нет сигнала разрешения «нашему» активному устройству). Предположим, что «наше» активное устройство запрашивает доступ к магистрали, выдавая сигнал R=0, а конкуренты отсутствуют. Сигнал R=0 вызывает формирование единицы в точке a и выдачу запросного сигнала BR=0 в общую для всех устройств линию BR. Сигнал a вызывает формирование сигнала $P_{ВЫХ}=0$, который, распространяясь по цепям $P_{ВХ}$—$P_{ВЫХ}$ последующих устройств, расположенных справа (на рисунке не показаны), запрещает обработку запросов в аналогичных схемах. С другой стороны, сигнал a=1, пройдя через три инвертора с RC-цепью, вызывает появление сигнала разрешения доступа к магистрали G=0.

Если после освобождения магистрали сигналы в точках a нескольких устройств одновременно приняли единичные значения, то

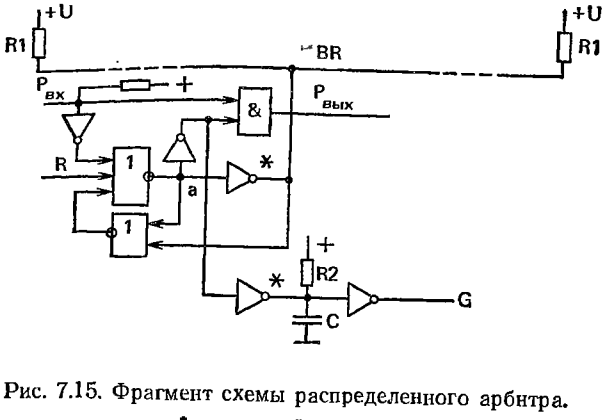


Рис. 7.15. Фрагмент схемы распределенного арбитра.
* — открытый коллектор

«победит» левое устройство в результате распространения сигналов $P_{вх}$ — $P_{вых}$ по приоритетной цепочке. Кратковременные положительные импульсы в точках а «проигравших» устройств будут подавлены соответствующими RC-цепями. Задержка, обеспечиваемая RC-цепью, должна превышать задержку распространения приоритетного сигнала по суммарной цепи — от первого устройства до последнего. Совокупность схем, показанных на рис. 7.15, образует единый арбитр, наращиваемый по мере увеличения числа активвых устройств.

7.13. Четырехпроводная магистраль для двунаправленной передачи информации в вычислительной системе [83, 130]

На рис. 7.16 показана схема связи блоков B_i (i=1, 2,..., n), входящих в вычислительную систему, по четырехпроводной магистрали М. На схеме показаны провода для передачи синхронизирующих сигналов CL и даниых D. Каждый сигнальный провод скручен со «своим» земляным проводом. Блок B_i содержит центральный узел H_i (процессор, память, контроллер и т. д.), два приемных магистральных элемента F и два передающих элемента VT. Элементы F должны иметь высокое входное сопротивленне. Элементы VT могут быть выполнены на биполярных транзисторах. Сигналы в магистрали представлены в прямом коде — лог. 1 соответствует высокий уровеиь напряжения в линии (все транзисторы VT, подключенные к линии, выключены), а лог. 0 — низкий (включен хотя бы один транзистор VT).

Вычислительная система может содержать, например, в качестве блоков B_i 3 процессора, 10 блоков памяти, 15 контроллеров, в том числе каналы прямого доступа, и другие устройства. Поэтому возможны конфликты, связанные с одновременной попыткой не-

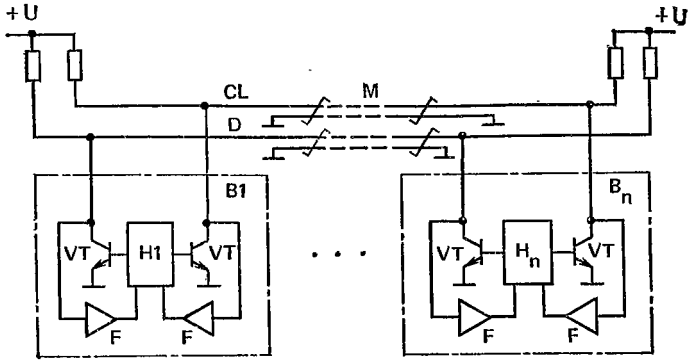


Рис. 7.16. Четырехпроводная магистраль

скольких устройств завладеть магистралью и провести сеанс связи с нужным абонентом. Разрешение таких конфликтов будет рассмотрено далее, а пока предположим, что конфликтов нет.

Процесс передачи данных между двумя устройствами в установившемся режиме показан на рис. 7.17, *а*. Смена данных на линии D допускается в те периоды времени, когда CL=0. При CL=1 данные D должны быть стабильными. На рис. 7.17, *б* показаны две особые ситуации, характеризующие начало и конец работы активного устройства на магистрали. Начало его работы характеризуется переходом сигнала D из 1 в 0, в то время как на линии CL поддерживается сигнал лог. 1. Такая ситуация, как видно из рис. 7.17, *а*, не встречается в режиме передачи данных, и, следовательно, ее нетрудно распознать. После выдачи признака начала сеанса связи следует пересылка информационных бит в соответствии с рис. 7.17, *а*. На рис. 7.17, б условно показана передача единственного бита x, равно-

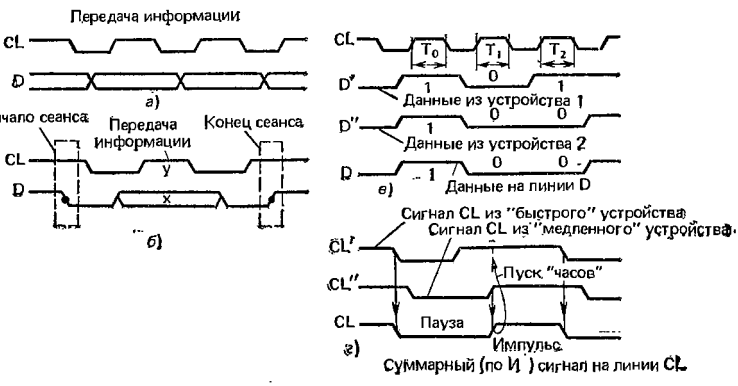


Рис. 7.17. Временны́е диаграммы работы магистрали

го нулю или единице в период прохождения синхроимпульса у. Конец сеанса связи характеризуется переходом сигнала D из нуля в единицу при наличии стабильной единицы на линии CL. Такая ситуацвя, как видно из рис. 7.17, *а*, также не встречается в режиме передачи данных.

В отсутствие конфликтов или после их разрешения такая система связи позволяет производить передачу адресной, командой и числовой информации в ту или иную сторону. Все устройства, например, могут «знать», что после признака начала сеанса связи последует служебный байт (для разрешения конфликтов), затем 20 адресных и 16 командных бит, затем по магистрали будут передаваться данные под управлением сигналов CL из ведущего или ведомого блока. В течение одного и того же сеанса связи инициатива по управлению магистралью может несколько раз переходить от ведущего устройства к ведомому и наоборот для подтверждения факта полного «взаимопонимания» между устройствами или для других целей (для обмена контрольными суммами, байтами состояния и т. д.).

Конец сеанса связи характеризуется появлением уже описанного сочетания сигналов на линиях CL и D. При правильной работе в дальнейшем должен вновь последовать признак начала сеанса и т. д. На протяжении сеанса связи i-го устройства с j-м k-е устройство, готовое стать ведущим, вынуждено ждать признака конца текущего сеаиса. Это устройство «знает», что предыстория системы такова, что признак начала связи уже имелся к тому моменту, когда оно (k-е устройство) пожелало воспользоваться магистралью, а признака конца еще нет. В таком состоянии ожидания могут находиться сразу несколько устройств. Поэтому после появления признака конца связи признак начала нового сеанса может быть выдан одновремеино (или почти одновременно) из нескольких устройств-претендентов на пользование магистралью.

Для выявления победителя в конкурентной борьбе используется уже известный нам метод поразрядного анализа собственных адресов устройств-конкурентов (см § 2.5, 3.2). Каждое устройство «знает» собственный адрес и «видит» на линии D результат взаимодействия всех конкурентов. Сначала в линию D из всех устройств-претендентов выдается старший разряд адреса. Побеждают те устройства, которые выдали в эту линию 0. (Магистраль реализует функцию Монтажное И, т. е. лог. 0, представленный низким уровнем напряжения, имеет приоритет перед лог. 1.) Далее устройства, прошедшие первый этап отбора, выдают в линию D следующий разряд и оценивают свои шансы на дальнейшую борьбу и т. д. Таким образом, побеждает устройство с меньшим собственным адресом.

На рис. 7.17, *в* показан фрагмент процесса взаимодействия двух конкурирующих устройств. В период времени T_0 из обоих устройств выданы единицы (старшие разряды собственных адресов), на общей линии D присутствует единица, оба устройства продолжают конкурентную борьбу. В период времени T_1 оба устройства выдают в линию D нули (следующий разряд адреса). В период времени T_2 первое устройство «видит» в линии D нуль (выданный вторым устройством) и, зная, что оно само пытается выдать единицу в эту линию, выбывает из конкурса.

Для правильной работы механизма конкурсного отбора нужна взаимная синхронизация устройств. Из рис. 7.17, *г* следует, что такая синхронизация имеется. Линия CL, как и линия D, реализует функ-

цию Моитажное И, поэтому справедливы следующие утверждения.

1. Пауза между импульсами начинается с исчезновения единичного сигнала CL у «быстрого» устройства и заканчивается формированием следующего единичного сигнала у самого «медленного» из устройств, вступивших в борьбу. До тех пор пока это медленное устройство не подготовит очередной бит данных, не возникнет единица на общей линии CL, так что все конкуренты, даже «нетерпеливые», вынуждены ждать медленного участника взаимодействия и запускать свои часы по положительному фронту сигнала на линии CL.

2. Длительность импульса на линии CL определяется от момента «подъема» уровня сигнала медленным устройством до его «снятия» наиболее быстрым устройством. (Это не противоречит диаграмме передачи данных, показанной на рис. 7.17, *а*.)

В процессе отбора вместо собственных адресов устройств-конкурентов можно использовать адреса ведомых устройств, с которыми эти конкуренты собираются обмениваться информацией. Это позволяет исключить служебный байт из структуры передаваемого сообщения и вести арбитраж непосредственно при выдаче в линию D адреса ведомого устройства. Предположим, например, что три процессора желают обратиться к трем ячейкам памяти: первый процессор — к ячейке 35, второй — к ячейке 1029, третий — к ячейке 108. Все три процессора на начальном этапе считают себя ведущими, однако по мере выдачи адресных бит сначала отключается второй процессор, затем — третий и в конечном счете только первый процессор выполниет предписанную ему операцию чтения или записи, так как код 35 — наименьший из трех конкурирующих кодов. Два других процессора вступят в борьбу по окончании текущего сеанса связи, как уже было описано.

Если несколько процессоров обратились к одной и той же ячейке в режиме чтения, то они читают одну и ту же информацию, «не подозревая» об одновременности. Каждый процессор считает, что операцию чтения выполняет только он. (При работе с программным семафором, см. § 2.1, такой режим недопустим, так как оба процессора одновременно сочтут общий ресурс незанятым и приступят к пользованию им.) Если несколько процессоров обратились к одной и той же ячейке в режиме записи и записываемое число одинаково для всех процессоров, то такая запись выполняется. Если числа разные, то поразрядный арбитраж распространяется и на них: записывается меньшее из конкурирующих чисел. Если несколько процессоров пытаются одновременно произвести операции записи и чтения содержимого одной и той же ячейки, то поразрядный арбитраж распространяется и на коды команд: «побеждает», иапример, код команды чтения как имеющий меньшее числовое значение, чем код записи.

Формат кодовой посылки от ведущего устройства может предусматривать «широковещательные» команды, адресуемые ко всем абонентам одновременно. Такими командами могут являться, например, команды общего сброса, переключения на аварийный режим и др.

7.14. Перемещение свободных областей памяти в конечную ее часть без приостановки выполнения рабочих программ [107]

При решении многих задач, например при обработке графической информации, используется сегментирование программ и данных, расположенных в ОЗУ микроЭВМ. По мере работы системы некоторые сегменты «устаревают» и в памяти появляются «дырки», число которых со временем возрастает. Размеры сегментов (в отличие от размеров страниц при страничной адресации) неодинаковы. Это, с одной стороны, приводит к более эффективному использованию памяти, а с другой—создает трудности при уничтожении дырок. Обычно уничтожение дырок выполняется программным путем. При этом выполнение рабочих программ приостанавливается и процессор начинает плотно сдвигать оставшиеся полезные сегменты информации к начальным зонам ОЗУ, или, другими словами, процессор выталкивает свободные сегменты в конец ОЗУ, чтобы там образовалась непрерывная свободная область. После такого сдвига выполнение рабочих программ продолжается. В процессе сдвига микроЭВМ практически лишена возможности эффективно обрабатывать поток новой информации, что является недостатком таких систем.

Предлагается, не дожидаясь момента, когда число дырок достигнет критического значения, «рассасывать» их в процессе выполнения рабочих программ, используя для этого любые, даже самые незначительные приостановки выполнения рабочих программ (например, периоды ожидания прерываний).

Рассмотрим фрагмент памяти (с ее содержимым), показанный на рис. 7.18, *а*, *б* и *в* — соответственно до, в процессе и после перемещения. Предполагается, что сегменты A, C и D жизнеспособны, а сегменты B и E подлежат уничтожению. Предлагается следующая последовательность действий. Сегмент B сначала меняется местами с сегментом C, а затем, на последующих этапах (на рисунке не показаны) — с сегментом D, затем двойной сегмент B—E как единое целое перемещается вниз. В результате все свободные сегменты образуют непрерывную область в конечной части ОЗУ, а все рабочие — непрерывную область в начальной части ОЗУ. Схема аппаратной части этого механизма показана на рис. 7.19.

Процессор, желая перейти от распределения памяти, показанного на рис. 7.18, *а*, к распределению, показанному на рис. 7.18, *в*, загружает регистр ширины зоны B (PrN), регистр-указатель конечного положения конца пустой зоны B после перемещения (PrX) и регистр-указатель текущей границы (PrY), которая по мере отработки задания опускается вниз. После того как регистры PrN, PrX и PrY загружены, процессор по праву считает, что работа по перемещению сегментов уже выполнена, хотя она, возможно, и не начиналась.

Логика работы схемы такова. Если адрес, поступающий из процессора при обращении к памяти через регистр РгАдр, лежит выше (геометрически, см. рис. 7.18, *б*) текущей границы Y, т. е. если код в регистре РгАдр меньше, чем код в регистре PrY, то соответствующая ячейка уже «подпрыгнула вверх» из зоны C″ в зону C′ (либо она относится к зоне A или к еще более «высокой» зоне) и, следовательно, адрес не требует преобразования. В этом случае на выходе h компаратора формируется единичный сигнал, который, пройдя через средний (по схеме) элемент ИЛИ, открывает путь непосредст-

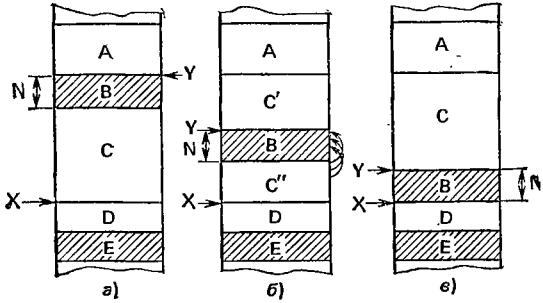


Рис. 7.18. Три стадии процесса ликвидации «дырки» в ОЗУ

венной передачи адреса из регистра РгАдр на выход схемы, т. е. к блоку ОЗУ.

Если адрес, поступающий из процессора, совпадает или находится ниже (геометрически) границы Y, но выше X, то соответствующая ячейка лежит в зоне C'' и для доступа к ней нужно к содержимому регистра РгАдр прибавить содержимое регистра PrN, что и делается с помощью сумматора. Если выданный процессором адрес лежит ниже X, то формируется сигнал L и адрес проходит в ОЗУ без изменений.

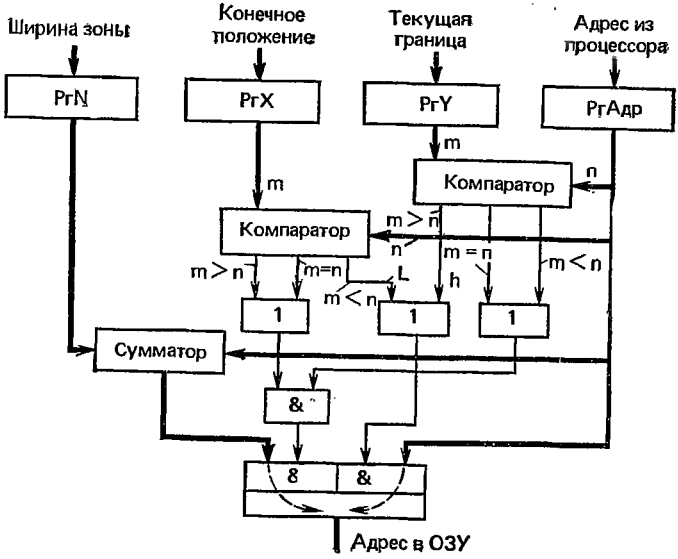


Рис. 7.19. Схема преобразования адресов

Как отмечалось, перемещение информации из зоны С″ в зону С′ идет с разделением времени между рабочей (высокоприоритетной) программой и сервисной программой перемещения (имеющей низкий приоритет). Эта сервисная программа по мере пересылки содержимого ячеек из С″ в С′ прибавляет единицу к содержимому регистра РгY и проверяет условие окончания работы (число перемещений должно быть равно N). Процессор, выполняя рабочую программу, «не подозревает» о том, что работа по перемещению еще не выполнена. Это, одиако, не приводит к «недоразумениям», так как схема создает «видимость оконченной работы» и отвечает за правильность преобразований.

7.15. Сокращение числа выводов однокристальной микроЭВМ за счет преобразования адресных сигналов [108]

На рис. 7.20 показано, как к микроЭВМ, содержащей внутреннее ОЗУ емкостью 2К ячеек (000—7FF), подключить внешнее ОЗУ емкостью 4К ячеек, «подстыковав» его в адресном пространстве к внутреннему ОЗУ. При этом число связей между обоими устройствами минимально. Инвертор D4 может быть исключен из схемы. В этом случае линия A11 соединяется непосредственно с линией A′11. На рис. 7.21, *а* и *б* показаны соответствия инутренних и внешних адресных пространств для случая с включением инвертора D4 и непосредственного подключения линии A11 к внешнему ОЗУ.

Рассмотрим, например, второй случай. Если процессор обращается по адресам 0—(2К—1), то он попадает в собственное ОЗУ. Элемент Исключающее ИЛИ D3 вырабатывает нулевой сигнал

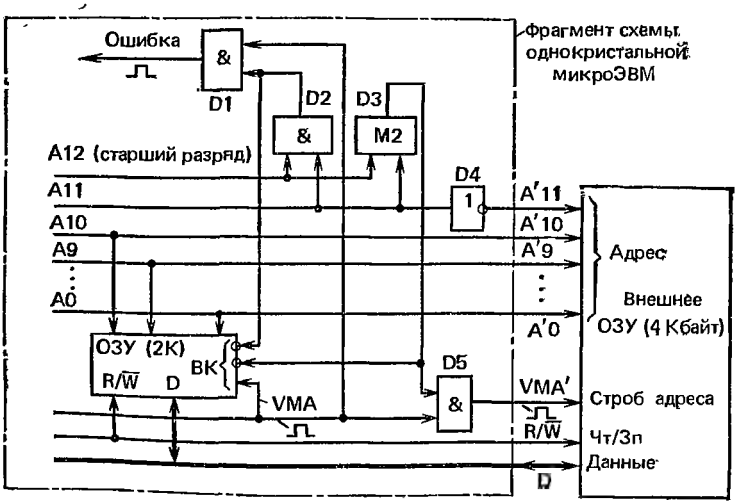


Рис. 7.20. Схема связи микроЭВМ с внешним ОЗУ

(0⊕0=0), открывающий по инверсному входу выбора кристалла ВК внутреннее ОЗУ (2К ячеек) и в то же время препятствующий передаче сигнала подтверждения истинности адреса VMA в линию VMA′. Поэтому внешнее ОЗУ оказывается нечувствительным к адресным сигналам на линиях A′0—A′11.

Если процессор обращается во вторую (считая сверху от единицы) зону объемом 2К ячеек своего адресного пространства, то A12=0, A11=A′11=1, следовательно с точки зрения внешнего ОЗУ обращение производится во вторую его половину (старший разряд

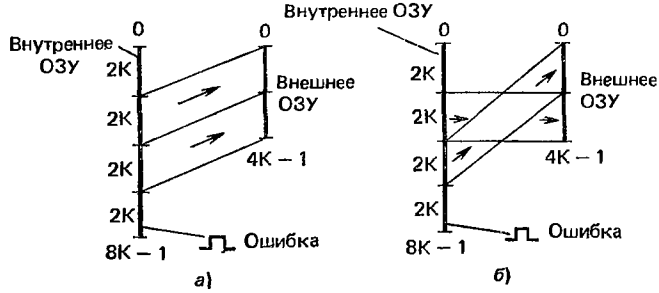


Рис. 7.21. Диаграммы соответствия адресных пространств:

а — при использовании инвертора D4; *б* — при непосредственном соединении адресной линии A11 с линией A′ 11 (без инвертора)

адреса равен 1). Элемент D3 на этот раз вырабатывает единичный сигнал (0⊕1=1), который закрывает внутреннее ОЗУ и открывает элемент И D5 по верхнему входу. Если процессор обращается в третью зону своего адресного пространства, то A12=1, A11=A′11=0 и во внешнем ОЗУ выбирается ячейка из первого его полублока (старший разряд адреса равен 0). На выходе элемента D3 и в этом случае формируется сигнал лог. 1.

Наконец, если процессор в результате сбоя, ошибки программиста или по какой-либо другой причине пытается обратиться в последнюю зону размером 2К ячеек своего адресного пространства (она не занята), то срабатывает элемент И D2 и в процессор через элемент И D1 посылается сигнал уведомления об ошибке. Содержимое обоих ОЗУ, как видно из схемы, не может быть искажено, поскольку внутреннее ОЗУ закрывается по верхнему входу ВК, а внешнее не получает сигнал VMA′ из-за того, что элемент И D5 закрыт по верхнему входу.

7.16. Уменьшение числа линий адресной шины при подключении ОЗУ к процессору [131]

Предлагается сократить число линий адресной шины, соединяющей ЦП с ОЗУ (рис. 7.22) за счет передачи младших разрядов A0—A15 адреса параллельным кодом, а старших A16—A31 разрядов — последовательным. Для преобразования параллельного кода

в последовательный используется сдвиговый регистр RG1, для обратного преобразования — сдвиговый регистр RG2. Шина данных и линии шины управления, не имеющие отношения к существу предлагаемого решения, на рис. 7.22 не показаны.

При выдаче полного 32-разрядного адреса из процессора регистр RG1 находится в режиме приема информации, так как на вход P/S этого регистра подан сигнал лог. 1 (рис. 7.23). По первому положительному фронту сигнала CL1 группа старших разрядов адреса

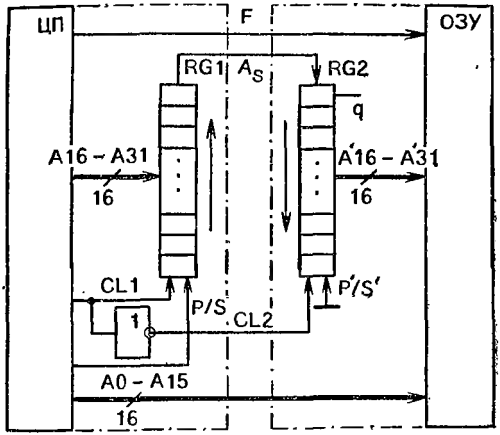


Рис. 7.22. Схема связи центрального процессора (ЦП) с ОЗУ.

Старшие разряды адреса (A16—A31) передаются последовательным кодом из сдвигового регистра RG1 в сдвиговый регистр RG2

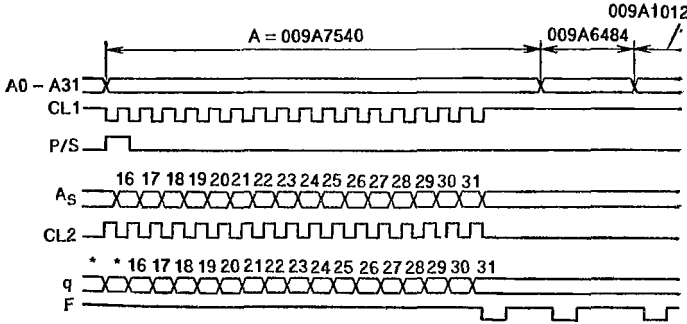


Рис. 7.23. Временные диаграммы передачи адреса страницы из процессора в ОЗУ.

Последующие обращения процессора к странице 009A не сопровождаются работой сдвиговых регистров

заносится в регистр RG1. После этого на вход P/S этого регистра подается сигнал лог. 0, регистр переводится в режим сдвига информации. Далее производится серия сдвигов, в результате старшие разряды адреса A16—A31 последовательным кодом передаются в регистр RG2.

После окончания передачи старших разрядов адреса процессор формирует сигнал лог. 0 на линии F. Этот сигнал подтверждает истинность адреса, принятого в регистр RG2; ОЗУ выполняет предписанную операцию считывания или записи данных в ячейку с адресом, равным, например, $009A7540_{16}$. Два последующих обращения процессора к ОЗУ, как показано в качестве примера на рис. 7.23, производятся при тех же значениях старших разрядов адреса (009A). Поэтому нет необходимости повторных передач старших разрядов адреса по линии A_S и процессор формирует сигналы F, подтверждая истинность установленного в регистре RG2 кода. При переходе с одной страницы (64К ячеек) на другую последовательная передача возобновляется.

Если регистр RG1 является составной частью микросхемы процессора (см. штриховые линии на рис. 7.22), то число внешних выводов микросхемы, требуемых для передачи старших разрядов адреса, уменьшается от 16 до 3 (A_S, CL2, F). Регистр RG2 целесообразно ввести в аппаратуру ОЗУ для уменьшения числа связей с процессором.

Чтобы уменьшить время передачи последовательного кода, можно использовать вторую пару сдвиговых регистров и вместо передачи одной группы из 16 разрядов передавать две группы по 8 разрядов. Число линий управления при этом не увеличивается, так как обе пары регистров работают под общим управлением.

Если программа и данные размещены на разных страницах памяти (каждая страница содержит 64К ячеек), то процессор при обращении за командами и при считывании или записи данных будет вынужден часто менять код в регистре RG2. Это связано с потерями времени, которые, однако, можно уменьшить, если учесть, что коды в регистре RG2 хотя и меняются, но в основном повторяются. Действительно, процессор обычно (по крайней мере, в достаточно малых интервалах времени) работает с двумя—тремя страницами (областями программ и данных) памяти и, следовательно, в регистре RG2 будут присутствовать, периодически сменяя друг друга, два—три кода. Поэтому целесообразно запоминать номера страниц, к которым в предыстории производилось обращение, и по возможности использовать эту информацию для исключения лишних последовательных передач.

На рис. 7.24 показана схема связи процессора с ОЗУ, содержащая два (вместо одного) регистра-приемника последовательного кода старших разрядов адреса. Выбор одного из двух регистров RG2 или RG3 производится сигналом на линии SEL: при выдаче процессором сигнала лог. 0 в эту линию активизируется регистр RG2, при выдаче сигнала лог. 1 — регистр RG3.

При первоначальном обращении процессора к ячейке памяти на странице X номер этой страницы последовательным кодом передается по линии A_S и запоминается в регистре RG2. Если процессор далее обращается к странице Y, то номер этой страницы последовательным кодом передается в регистр RG3. При повторных обращениях к страницам X и Y уже нет необходимости загружать регистры

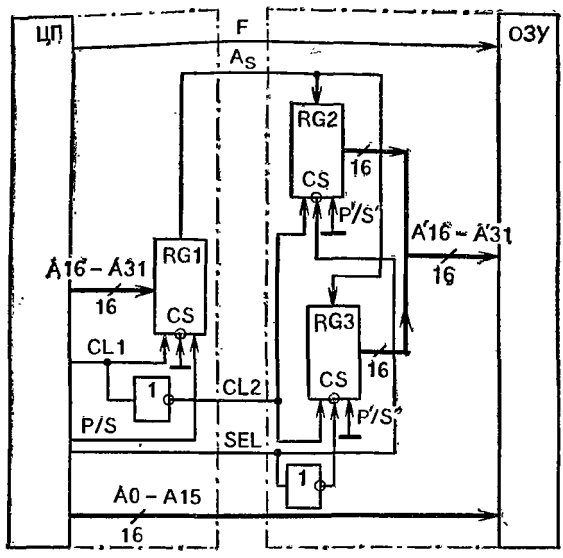


Рис. 7.24. Использование двух регистров-приемников позволяет хранить одновременно два номера страниц, с которыми работает процессор, и оперативно выбирать любую из этих страниц

RG2 или RG3 — в них имеется нужная информация. Поэтому процессор выбирает нужный регистр и формирует сигнал F подтверждения истинности адреса. Аппаратура процессора при таком способе адресации памяти усложняется. Процессор должен сравнивать текущие адреса с содержимым копий регистров RG2 и RG3 и по результатам сравнения либо передавать последовательный код по линии A_S в регистр с более «старой» информацией (которая, может быть, уже не нужна), либо без лишних затрат времени выбирать нужный регистр и формировать сигнал лог. 0 на линии F. Число регистров-приемников последовательного кода может быть увеличено до 3—4.

7.17. Компактное размещение пользовательских адресных пространств в общей физической памяти [109]

Предположим, что имеются несколько независимых пользователей одной микроЭВМ. Каждый пользователь считает, что в его распоряжении находится адресное пространство емкостью 64К ячеек, каждой из которых соответствует ячейка памяти (рис. 7.25). Во всех адресных пространствах имеется область Q системных программ, которые обеспечивают пользователю необходимый «сервис». Задача состоит в том, чтобы при суммировании всех пользовательских адресных пространств в общей физической памяти избежать много-

кратного повторения зон Q, поскольку они одинаковы по содержанию. В то же время каждый пользователь должен считать, что в его распоряжении находится адресное пространство, не связанное с другими адресными пространствами.

Суть предложения заключается в том, что в реальном (физическом) адресном пространстве общий блок Q записан только в одном месте (в начале), а программы и данные пользователей (блоки M_i) расположены последовательно, вплотную друг к другу, как показано на рис. 7.26, *а*. Схемное решение приведено на рис. 7.26, *б*.

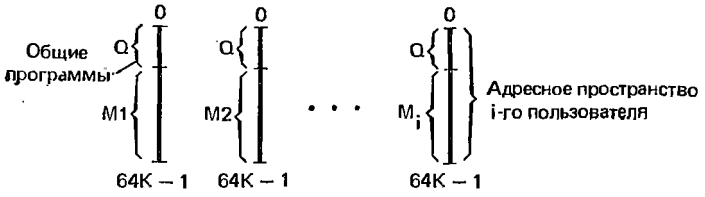


Рис. 7.25. Адресные пространства пользователей

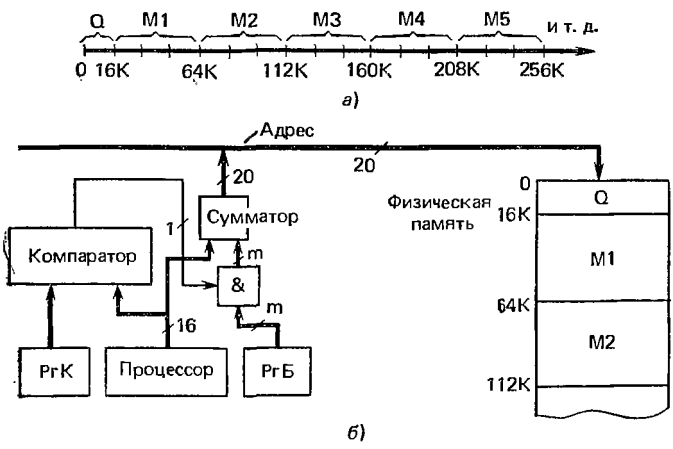


Рис. 7.26. Схема преобразования адресов

Программы пользователей поочередно вступают в фазу активности, например, через каждую миллисекунду. Каждой программе выделен определенный код базы, определяющий физический адрес начала соответствующего блока M_i. В регистре константы (РгК) задан адрес, равный граничному адресу блока Q. Код базы загружается процессором в программно-доступный регистр базы пользователя (РгБ).

Если адреса, поступающие в дальнейшем из процессора, не превышают по величине заданный порог (константу), то компаратор не

срабатывает, элементы И закрыты, поэтому 16-разрядный адрес, поступающий из процессора, проходит через сумматор без изменения и поступает в память. Этот адрес относится к одной из ячеек общего блока Q. Если из процессора поступил адрес, превышающий порог, то компаратор срабатывает и к этому 16-разрядному адресу арифметически прибавляется содержимое регистра РгБ. В результате происходит «скачок» адреса в ту область памяти, которая отведена данной пользовательской программе.

7.18. Схемное определение наиболее «устаревшего» блока буферной памяти [90, 198]

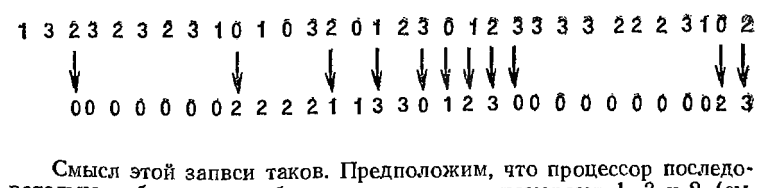
В § 1.5 при рассмотрении виртуальной памяти отмечалось, что наиболее естественно уничтожать в буферной памяти те блоки, которые дольше всего не использовались, когда возникает вопрос о том, куда поместить новую информацию. Основная схема определения блока-кандидата на уничтожение приведена на рис. 7.27, *а* [90].

Предположим, что буферная память содержит блоки 0, 1, 2 и 3 (на рисунке не показаны). При обращении к блоку 0 адресный дешифратор (не показан) формирует положительный импульс на своем выходе X0, при обращении к блокам 1—3 — импульсы на выходах X1—X3. Сигнал Y_i принимает единичное значение только тогда, когда дольше всего не было обращения к блоку i (i=0, 1, 2, 3).

Рассмотрим, например, условие формирования единичного сигнала на выходе Y1. Этим условием является наличие трех единиц на входах элемента И D5. Последнее означает, что триггер D1 установлен в 1 (сигнал X0 пришел позже, чем X1), триггер D2 — в 0 (сигнал X2 пришел позже, чем X1) и триггер D3 — в 0 (сигнал X3 пришел позже, чем X1). Как видим, Y1=1 только тогда, когда сигналы X0, X2 и X3 поступили позже (в любой последовательности), чем X1.

Аналогично можно убедиться в корректности формирования остальных сигналов.

Далее в качестве примера приведена некоторая последовательность номеров блоков буферной памяти, к которым обращается процессор (верхняя строка), и последовательность номеров блоков-кандидатов на уничтожение (нижняя строка):



Смысл этой записи таков. Предположим, что процессор последовательно обращался к блокам памяти с номерами 1, 3 и 2 (см. начало верхней строки). Блок 0 в этом случае — самый «старый», поскольку в ближайшей предыстории к нему не было обращения, а ко всем остальным обращения были. Поэтому схема выбирает в качестве кандидата на уничтожение блок 0, что отражено стрелкой, направленной в начало второй строки. Просматривая верхнюю строку, приходим к комбинации «...310...», из которой следует, что

«устарел» блок 2 (см. вторую стрелку в приведенной записи) и т. д.

Схему можно каскадировать с целью увеличения числа обслуживаемых блоков.

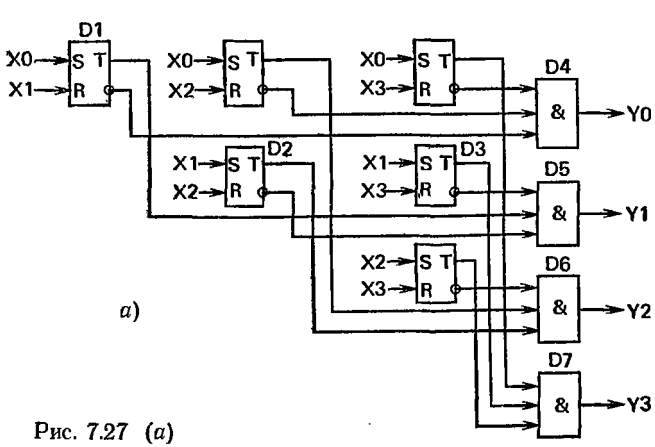
В схеме рис. 7.27, *б* [198] учтена возможность удаления блоков памяти, например, в связи с их ремонтом. При исключении блока i (i=0, 1, 2, 3) один из триггеров запрета, предварительно установленных в 0 сигналом Сброс, переводится в 1 под действием импульса $Запрет_i$. В результате соответствующий элемент И выключается по нижнему входу. Сигнал с единичного плеча триггера, соответствующего отключенному блоку, имитирует «недавнее обращение» к этому блоку памяти, создавая условия выбора одного из оставшихся блоков.

7.19. Варианты построения арбитров, реализующих гибкое обслуживание абонентов

В рассмотренной ранее схеме арбитра (см. § 2.2, рис. 2.13) использовалась решетчатая структура, в которой каждый канал встречается с каждым в одной из 28 узловых схем. В зависимости от значений 28 управляющих сигналов, определяющих настройку узловых схем, возможны 2^{28} режимов работы, среди которых имеется 8! режимов жесткого задания приоритетов и 2^{28}—8! гибких режимов.

Любой гибкий режим, как отмечалось, позволяет в большей или меньшей степени удаляться от абсолютного приоритетного порядка каналов. При этом «сильные» каналы в определенных ситуациях проигрывают слабым, т.е. в конкурентной борьбе нет абсолютного лидера.

Однако с увеличением интенсивности потока входных запросов поведение арбитра становится все более «жестким». В частности, при постоянном действии одновременно всех запросных сигналов постоянно (в каждом акте арбитража) обслуживается один и тот же канал. Действительно, если настройка узловых схем постоянна, то при постоянном наличии всех запросных сигналов однозначно



а)

Рис. 7.27 (*а*)

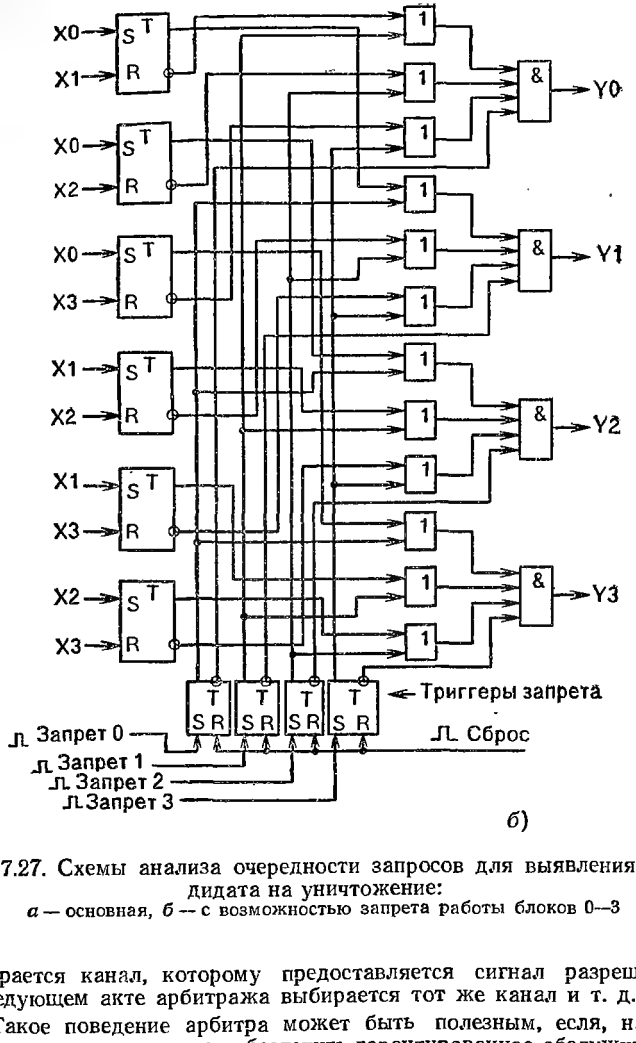


Рис. 7.27. Схемы анализа очередности запросов для выявления кандидата на уничтожение:
а — основная, *б* — с возможностью запрета работы блоков 0—3

выбирается канал, которому предоставляется сигнал разрешения. В следующем акте арбитража выбирается тот же канал и т. д.

Такое поведение арбитра может быть полезным, если, например, в «часы пик» следует обеспечить гарантированное обслуживание одного из каналов даже в ущерб оставшимся. Однако можно представить себе и противоположные ситуация, когда в критические периоды демократичность обслуживания должна сохраняться и никто не должен стать безусловным лидером.

Чтобы сохранить демократичность обслуживания каналов при пиковых интенсивностях потоков входных запросов, можно динамически менять настройку узловых схем. Одновременно с подачей на информационные входы арбитра очередного запросного кода на уп-

равляющие входы поступает очередной код настройки, который остается неизменным на протяжении одного акта арбитража. В следующем акте принимается новая пара кодов и т. д. Рассмотрим сначала особенности работы арбитра (рис. 2.13) при равновероятном выборе победителя узловой схемой. Чтобы обеспечить такой выбор в каждой узловой схеме, можно использовать 28-разрядный генератор G псевдослучайных чисел, подключив его выходы Q1—Q28 к управляющим входам соответствующих узловых схем, рис. 7.28, *а* (входной и выходной регистры для фиксации сигналов запросов и разрешений на время выполнения акта арбитража, а также схемы тактирования не показаны для упрощения рисунка).

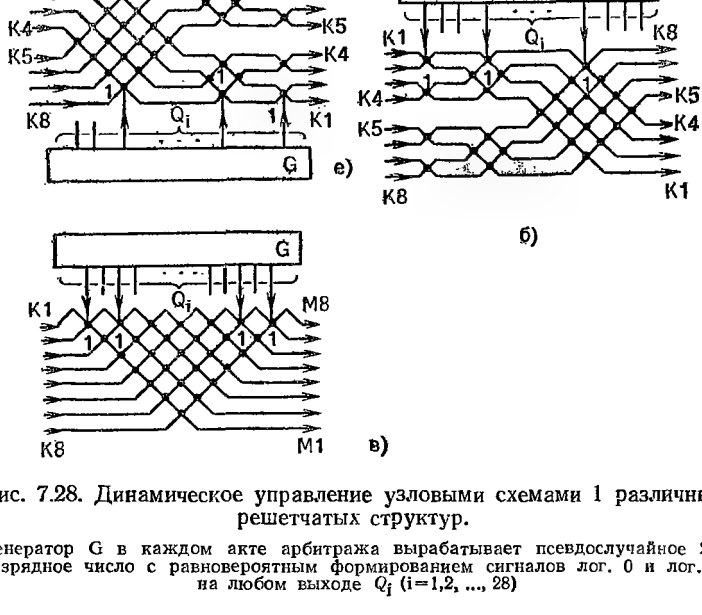


Рис. 7.28. Динамическое управление узловыми схемами 1 различных решетчатых структур.

Генератор G в каждом акте арбитража вырабатывает псевдослучайное 28-разрядное число с равновероятным формированием сигналов лог. 0 и лог. 1 на любом выходе Q_i (i=1,2, ..., 28)

В этой схеме (рис. 7.28, *а*) каналы К1—К8 оказываются неравноправными. Например, каналы К1 и К8 имеют несколько бо́льшие шансы выйти победителем в борьбе с конкурентами, чем каналы К4 и К5. Это связано с тем, что сигналы по каналам К1 и К8 сразу (на льготных условиях) попадают во внутренние ярусы решетчатой структуры и у них больше шансов дойти до выходов, чем у сигналов по каналам К4 и К5, которые, прежде чем достичь уровня вхождения сигналов по каиалам К1 и К8, подвергаются дополнительному риску быть уничтоженными при встрече с конкурентами. Такое не слишком значительное расслоение приоритетов каналов может рассматриваться как достоинство или недостаток устройства в завнсимости от нужд потребителя. Для выравнивания приоритетов каналов

достаточно развернуть решетчатую структуру на 180°, рис. 7.28, *б* (входы по-прежнему слева, выходы — справа). Напротив, для более существенного расслоения приоритетов можно использовать схему, приведенную на рис. 7.28, *в*.

В схеме рис. 7.28, *в* неравноправность каналов доведена до максимума за счет применения «треугольной» решетчатой структуры из тех же узловых схем, что и ранее (рис. 2.13, *б*). В данной структуре, как и в предыдущих (рис. 7.28, *а, б*), каждый канал встречается с каждым в одной из узловых схем, так что на выходы может прой-

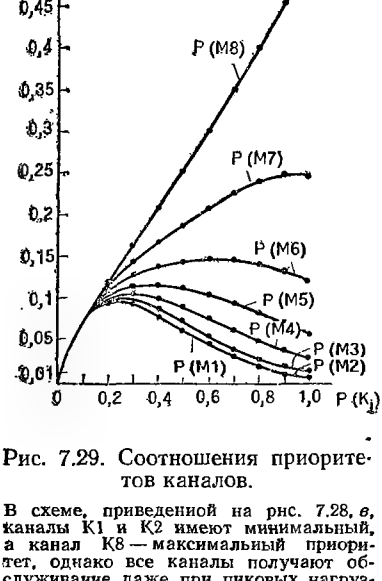


Рис. 7.29. Соотношения приоритетов каналов.

В схеме, приведенной на рис. 7.28, *в*, каналы К1 и К2 имеют минимальный, а канал К8 — максимальный приоритет, однако все каналы получают обслуживание даже при пиковых нагрузках, при $P(K_i) \to 1$

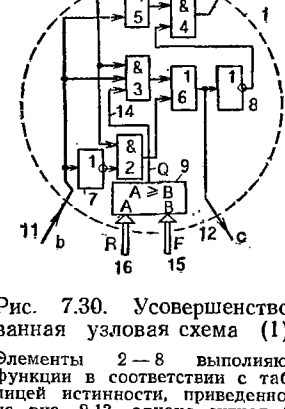


Рис. 7.30. Усовершенствованная узловая схема (1).

Элементы 2 — 8 выполняют функции в соответствии с таблицей истинности, приведенной на рис. 2.13, однако сигнал Q в каждом акте арбитража может быть равен 0 или 1 в зависимости от соотношения чисел F и R на входах 15 и 16 компаратора 9

ти только один сигнал независимо от значения управляющего кода и числа имеющихся запросных сигналов (при отсутствии запросных сигналов на выходах сформированы лог. 0).

Асимметрия приоритетов каналов обусловлена тем, что сигнал по каналу К1 испытывает на себе «свежие силы» противников (К2—К8) и имеет сравнительно мало шансов пройти на выход М1. Например, если имеются все конкурирующие сигналы, то возможность выигрыша реализуется только при благоприятной настройке всех семи узловых схем, встречающихся на пути распространения сигнала по первому каналу (К1—М1), что случается сравнительно редко, в среднем в одном случае из $2^7 = 128$.

В то же время сигнал по каналу К8 распространяется «под прикрытием» соседних каналов, которые берут на себя основную тя-

жесть конкурентной борьбы. Действительно, сигнал по каналу К8 в любом случае встречается на своем пути не более чем с одним конкурентом, которому в результате отбора удалось достичь правого катета треугольной матрицы. Исход поединка при этом равновероятен.

Приведенные рассуждения подтверждаются результатами моделирования работы устройства на микроЭВМ (ДВК-2М). На рис. 7.29 показаны графики зависимостей вероятностей появления сигналов на выходах разрешений устройства от вероятностей появления запросных сигналов на его входах. Для простоты предполагается, что вероятности появления сигналов на входах К1—К8 устройства одинаковы: $P(K1)=P(K2)=...=P(K8)=P(K_i)$. Верхняя кривая соответствует вероятности появления сигнала на выходе М8 ($P(M8)$), следующая — вероятности появления сигнала на выходе М7 ($P(M7)$) и т. д. Из рассмотрения диаграмм, приведенных на рис. 7.29, можно сделать следующие выводы.

1. Вероятность появления сигнала на выходе М8 ($P(M8)$) в любых случаях превышает вероятность появления сигнала на любом другом выходе.

2. Вероятности появления сигналов на выходах М1 и М2 $P(M1)$ и $P(M2)$ одинаковы и минимальны.

3. Вероятности появления сигналов на остальных выходах упорядочены в соответствии с нумерацией выходов.

4. При малых вероятностях появления запросных сигналов конфликты редки и вероятность появления сигнала на выходе практически равна вероятности появления запросного сигнала на соответствующем входе: $P(M8)\approx P(M7)\approx...\approx P(M1)\approx P(K_i)$. При $P(K_i)\to 0$ приближенные равенства стремятся к точным.

5. С увеличением вероятностей появления запросных сигналов начинается расслоение каналов на более приоритетные и менее приоритетные.

6. Восьмому и седьмому каналам наиболее выгодны пиковые ситуации, когда вероятности появления запросных сигналов $P(K_i)$ приближаются к единице. В этих ситуациях особенно заметно расслоение приоритетов каналов.

7. Для остальных каналов наиболее выгодны промежуточные значения $P(K_i)$. Например, шестой канал наиболее часто выигрывает при $P(K_i)\approx 0{,}6$.

8. Ни одна из диаграмм $P(M_i)$ не соприкасается с горизонтальной координатной осью, т. е. все каналы получают обслуживание даже в пиковых ситуациях, хотя в то же время имеется явное расслоение приоритетов каналов.

В схемах, приведенных на рис. 7.28, использовалась настройка всех узлов решетчатой структуры на равновероятный выбор победителя при конфликтах. Более «тонкие» способы настройки могут быть получены при использовании усовершенствованной узловой схемы, в которой (в отличие от схемы, приведенной на рис. 2.13, *б*) сигнал Q поступает не с триггера, а с компаратора (рис. 7.30).

Сигнал Q=1 на выходе 14 компаратора 9 формируется в тех случаях, когда случайное число R, поступающее на его входы 16, больше или равно числу F, поступающему на входы 15.

Случайное число R представлено n двоичными разрядами и является равномерно распределенным в диапазоне 0 — (2^n-1). На протяжении одного акта арбитража число R не изменяется, в сле-

дующем акте арбитража на входы 16 поступает новое случайное число и остается стабильным на протяжении этого акта и т. д.

Число F, представленное n двоичными разрядами, поступает на входы 15 компаратора 9 и задает порог его срабатывания. Это число, по существу, задает вероятность появления единичного сигнала (по шкале из 2^n градаций) на выходе Q. Например, если F=0, то Q=1 независимо от значения R, т. е. вероятность обнаружения единичного сигнала на выходе Q равна единице. Если, например, $F=2^n-1$, то сигнал Q=1 будет формироваться сравнительно редко, только в тех случаях, когда R=F, т. е. в среднем в одном случае из 2^n (вероятность обнаружения единичного сигнала на выходе Q составит $1/2^n$). При всех промежуточных значениях F (2^n-2 значения) вероятность P(Q) обнаружения единичного сигнала на выходе Q будет меняться в пределах $1/2^n<P(Q)<1$.

Число F, так же как и число R, не должно меняться в течение одного акта арбитража. Изменение значения F производится в связи с необходимостью перестройки приоритетной структуры устройства, если такая необходимость возникает.

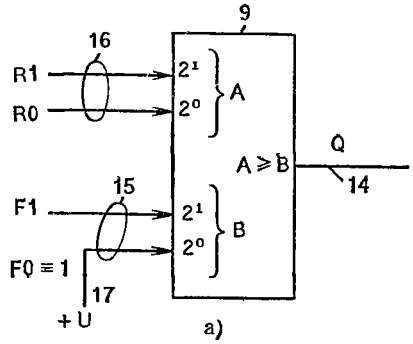
Узловая схема рис. 7.30 работает следующим образом. В очередном акте арбитража на входах 15 и 16 фиксируются числа F и R. На входы 10 и 11 поступают запросные сигналы. Если конфликта нет, то узловая схема выполняет функции двух пересекающихся, но изолированных друг от друга проводов 10—12 и 11—13. При наличии конфликта, в зависимости от соотношения чисел F и R, побеждает один из двух запросных сигналов. На этом акт арбитража заканчивается, и в дальнейшем процесс повторяется. Так как на входы 16 в каждом акте арбитража поступает очередное случайное число, поведение узловой схемы в общем случае также имеет случайный характер, в том смысле, что приоритеты каналов в каждом узле становятся вероятностными и низкоприоритетный канал при конфликтах не «оттесняется» высокоприоритетным, а периодически реализует шансы (пусть даже невысокие) на выигрыш.

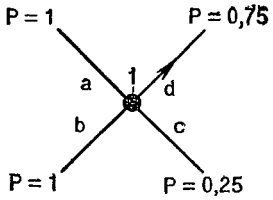
На рис. 7.31, *а* показан вариант использования компаратора двух 2-разрядных двоичных чисел для получения двух режимов вероятностной обработки сигналов, соответствующих рис. 7.31, *б* и *в*.

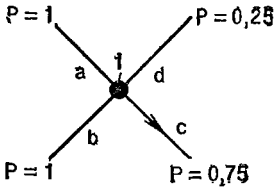
При F1=1 число B, воспринимаемое компаратором, равно 3, так как на вход F0 подано напряжение высокого уровня (+U), соответствующее сигналу лог. 1. Компаратор срабатывает только тогда, когда на его входы A также поступает число 3 (R0=R1=1, см. таблицу, приведенную на рис. 7.31, *б*).

Поведение узловой схемы при F1=1 можно интерпретировать так. Если сигналы a и b постоянно присутствуют на ее входах (вероятность появления каждого из этих сигналов равна единице), то в каждом акте арбитража исход конкурентной борьбы будет определяться соотношением чисел R и F, а именно в среднем в трех случаях из четырех Q=0 и будет побеждать направление b—d (вероятность прохождения сигнала на выход d составит $3/4=0{,}75$). В одном случае из четырех возможных (при R1=R0=1, см. таблицу, приведенную на рис.7.31, *б*) будет побеждать сигнал a. При этом вероятность прохождения сигнала на вход c равна $1/4=0{,}25$.

Вероятности появления входных и выходных сигналов для рассмотренного случая приведены на рис. 7.31, *б*, причем стрелка, исходящая из узловой схемы (обозначенной точкой), соответствует приоритетному направлению распространения сигналов при конфликтах. Такие же обозначения приняты на рис. 7.31, *в*, 7.32.







F1 = 1

R1	R0	Q
0	0	0
0	1	0
1	0	0
1	1	1

б)

F1 = 0

R1	R0	Q
0	0	0
0	1	1
1	0	1
1	1	1

в)

Рис. 7.31. Компаратор 2-разрядных чисел *(а)* и приоритетные структуры узловой схемы 1 (рис. 7.30), соответствующие условию F1=1 *(б)* и F1=0 *(в)*

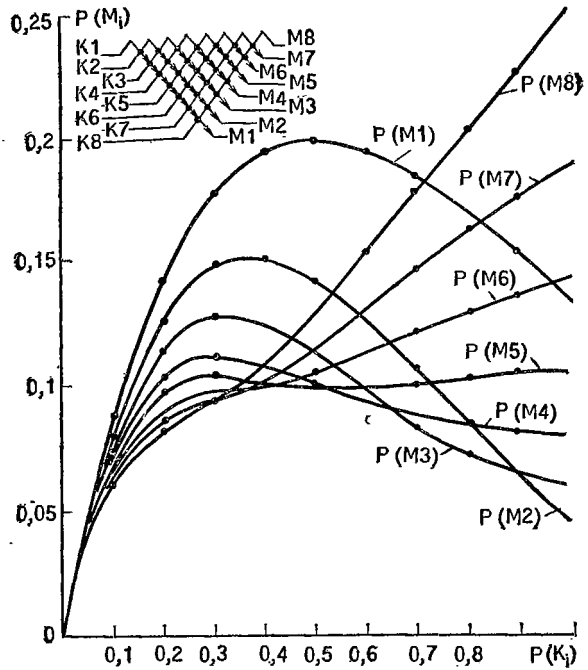


Рис. 7.32. Вероятности P (M_i) формирования единичных сигналов на выходах M_i устройства зависят от вероятностей P (K_i) появления входных сигналов.

Настройка узловых схем структуры вступает в противоречие со стремлением самой структуры оказать поддержку каналам с бо́льшими номерами. В результате с увеличением P(K_j) структура почти побеждает и ее останавливает лишь то, что значения P(K_j) не могут быть больше единицы

При F1=0 (см. рис. 7.31, *в*) B=1, что повышает вероятность формирования единичного сигнала Q до величины, равной 0,75, а это, в свою очередь, приводит к переключению стрелки с направления b — d на направление a — c (сравните рис. 7.31, *б* и *в*).

Таким образом, в данном примере с помощью единственного сигнала F1 из группы 15 можно управлять приоритетами каналов, обеспечивая в то же время демократичные отношения между ними, при которых возможные конфликты разрешаются в пользу того или иного канала.

Использование предлагаемой узловой схемы позволяет проектировать различные решетчатые структуры, реализующие вероятностные алгоритмы обработки входных сигналов (речь идет не только об арбитрах).

На рис. 7.32 показан вариант настройки уже встречавшейся ра-

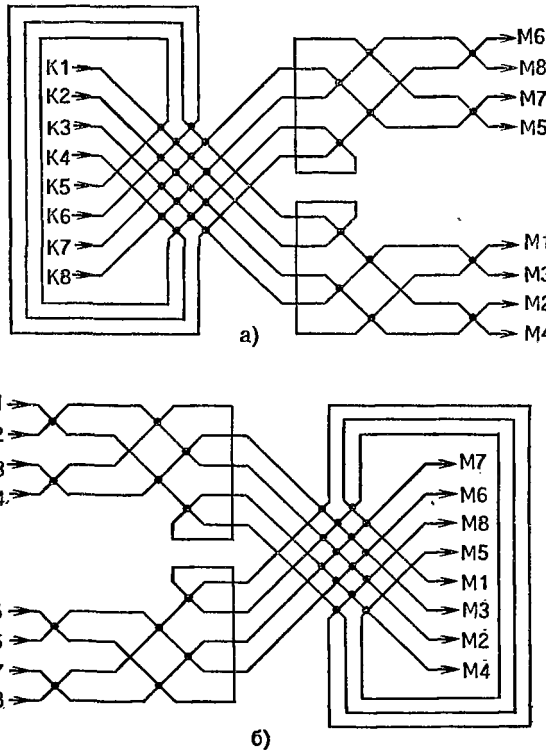


Рис. 7.33. Усовершенствованные решетчатые структуры арбитров.

Переход к трехмерным решеткам (они становятся однородными на цилиндрических поверхностях) позволяет повысить быстродействие арбитров и обеспечить равноправность каналов при управлении от генератора псевдослучайных чисел

нее «треугольной» структуры и соответствующие этому варианту настройки приоритетные зависимости. Направления стрелок, как отмечалось, соответствуют направлениям наиболее вероятной выдачи сигналов при конфликтах в узловых схемах ($P=0{,}75$, см. рис. 7.31, *в*). Из рисунка видно, что направления стрелок благоприятствуют приоритетному распространению сигналов по каналам с меньшими номерами. При $P(K_i)<0{,}3$ значения $P(M_i)$ упорядочены в соответствии с номерами каналов. Однако с увеличением вероятностей появления запросных сигналов приоритеты постепенно перераспределяются из-за того, что сама треугольная структура матрицы (выбранная в качестве примера) из узловых схем, как показано ранее, предоставляет более благоприятные условия каналам с бо́льшими номерами. В любых ситуациях все каналы получают обслуживание, а не «оттесняются» более сильными конкурентами.

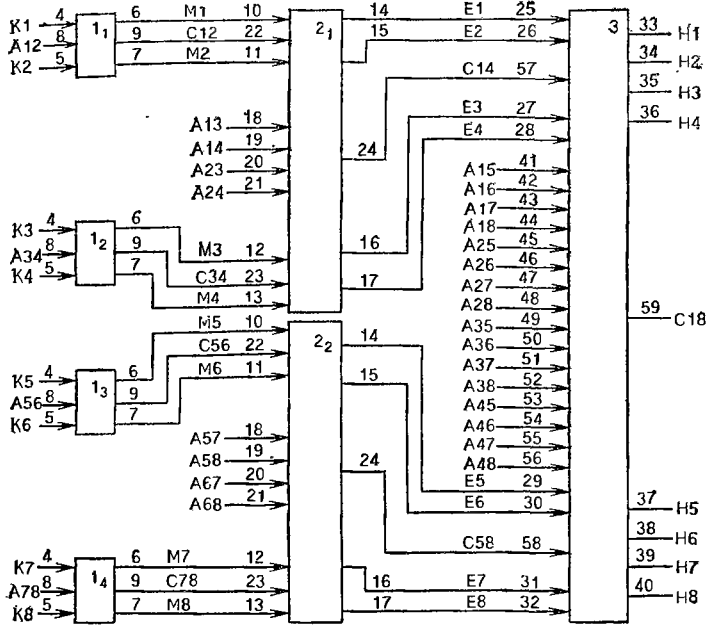


Рис. 7.34. Вариант схемы арбитра с повышенным быстродействием

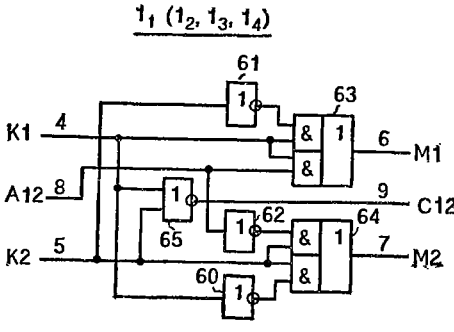


Рис. 7.35. Структура блоков 1_i (см. рис. 7.34)

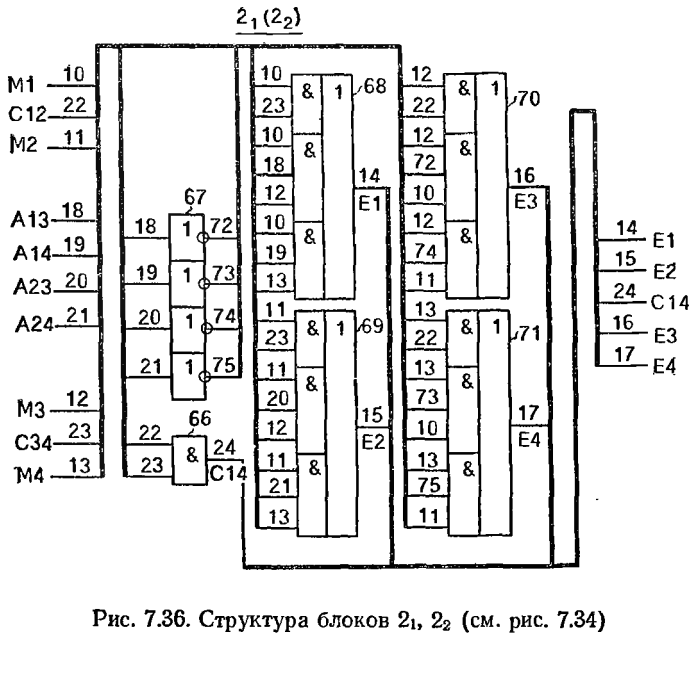


Рис. 7.36. Структура блоков 2_1, 2_2 (см. рис. 7.34)

В заключение рассмотрим усовершенствованные схемы арбитров.

На рис. 7.33 [205] показаны две схемы арбитров с равноправными каналами. В отличие от схемы, приведенной на рис. 7.28, *б*, суммарная задержка распространения сигналов от входов до выходов уменьшена до семи задержек узловых схем.

На рис. 7.34 [206] показан «распараллеленный» вариант схемы, приведенной на рис. 7.28, *б* (без учета генератора псевдослучайных чисел). Суммарная задержка распространения сигналов от входов до выходов уменьшена до четырех задержек логических элементов. Схема содержит блоки 1_1—1_4 первого уровня, блоки 2_1, 2_2 второго уровня и блок 3 третьего уровня, структура которых приведена на рис. 7.35—7.37. Управляющие сигналы A_{ij} (аналоги управляющих сигналов Q_i на рис. 7.28, *б*) определяют исход взаимодействия сигналов, поступающих одновременно по каналам K_i и K_j (только по этим каналам): при $A_{ij}=1$ побеждает сигнал по каналу K_i (формируется сигнал $H_i=1$), в противном случае, при $A_{ij}=0$, предпочтение отдается сигналу по каналу K_j ($H_j=1$). Сигнал C18 используется при увеличении числа обслуживаемых каналов.

На рис. 7.38 показан «распараллеленный» вариант схемы, приведенной на рис. 7.28, *в* (без учета генератора псевдослучайных чисел). Структура блоков 8, 1, 2, 3, 7 приведена на рис. 7.39—7.43. Сигналы A_{ij} имеют тот же смысл, что и в схеме, показанной на

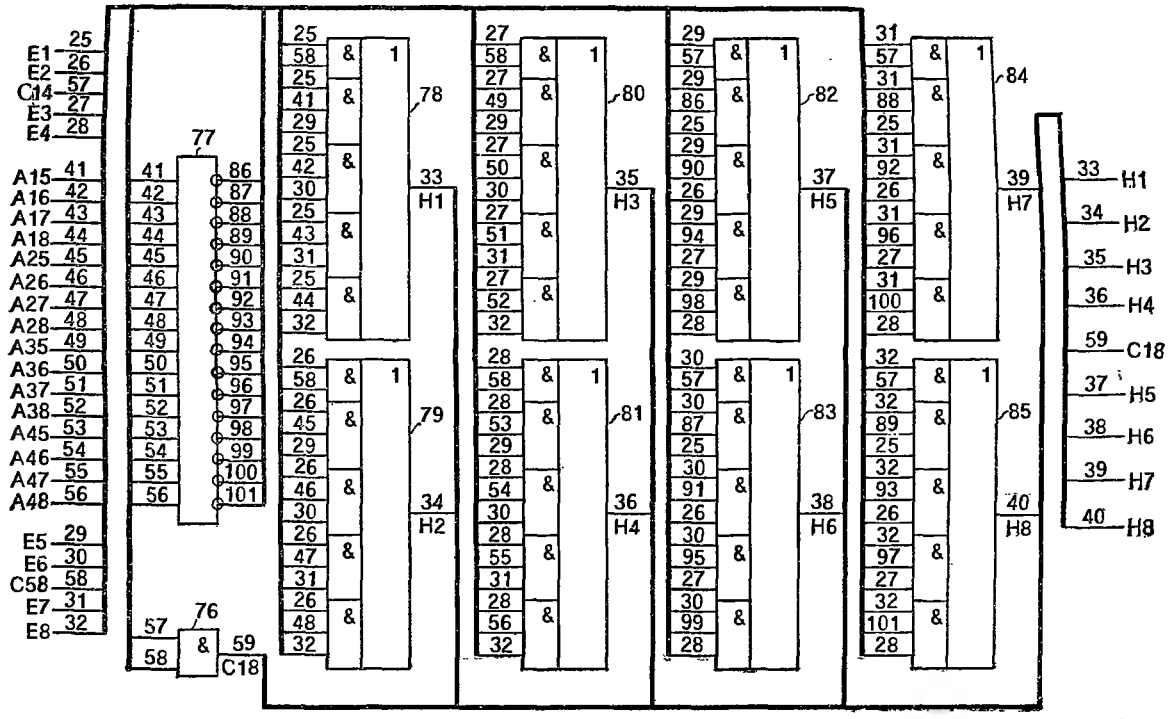


Рис. 7.37. Структура блока 3 (см. рис. 7.34)

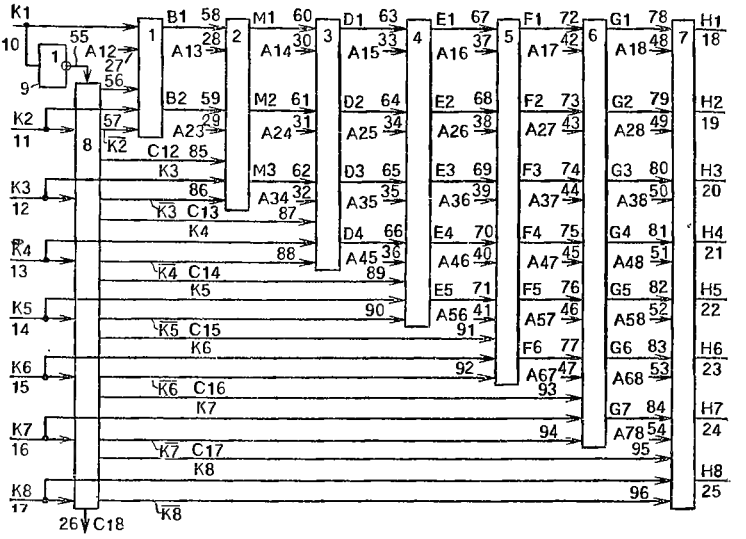


Рис. 7.38. Быстродействующий аналог треугольной матрицы из узловых схем (см. рис. 7.28, *в*)

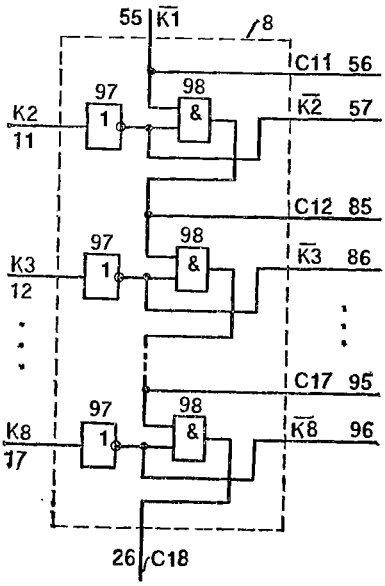


Рис. 7.39. Структура блока 8 (см. рис. 7.38)

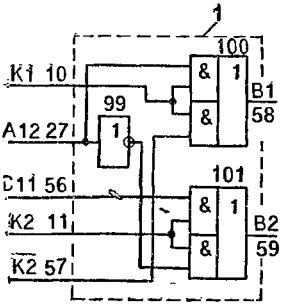


Рис. 7.40. Структура блока 1 (см. рис. 7.38)

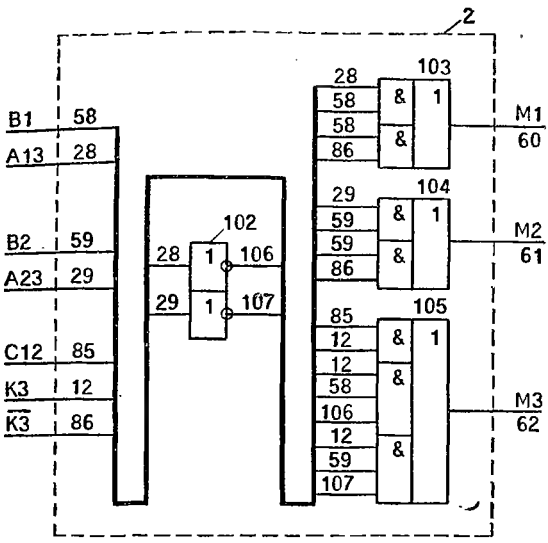


Рис. 7.41. Структура блока 2 (см. рис. 7.38)

рис. 7.34. Блоки 4—6 выполняются аналогично блокам 1, 2, 3, 7 и реализуют следующие функции.

$E1=D1(A15+\overline{K5})$;

$E2=D2(A25+\overline{K5})$;

$E3=D3(A35+\overline{K5})$;

$E4=D4(A45+\overline{K5})$;

$E5=K5(C14+D1\overline{A15}+D2\overline{A25}+D3\overline{A35}+D4\overline{A45})$;

$F1=E1(A16+K6)$;

$F2=E2(A26+\overline{K6})$;

$F3=E3(A36+\overline{K6})$;

$F4=E4(A46+\overline{K6})$;

$F5=E5(A56+K6)$;

$F6=K6(C15+E1\overline{A16}+E2\overline{A26}+E3A36+E4\overline{A46}+E5\overline{A56})$;

$G1=F1(A17+\overline{K7})$;

$G2=F2(A27+\overline{K7})$;

$G3=F3(A37+\overline{K7})$;

$G4=F4(A47+\overline{K7})$;

$G5=F5(A57+K7)$;

$G6=F6(A67+\overline{K7})$;

$G7=K7(C16+F1\overline{A17}+F2\overline{A27}+F3\overline{A37}+F4\overline{A47}+F5\overline{A57}+F6\overline{A67})$.

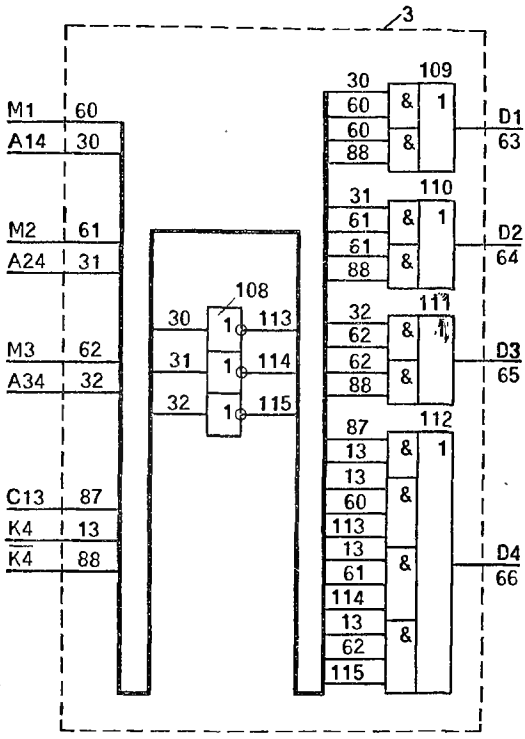


Рис. 7.42. Структура блока 3 (см. рис. 7.38)

7.20. Устройство для быстрого стирания информации в памяти микроЭВМ [132, 163]

При работе микропроцессорных систем часто возникает задача быстрого стирания информации в оперативной памяти, например, для «очистки» экрана дисплея, для обеспечения контроля правильности выполнения программы (см. § 6.6) и т. п. В системах коллективного пользования стирание бывает необходимо после выполнения текущей пользовательской программы, когда ОЗУ вместе с его старым содержимым переходит в распоряжение «чужой» пользовательской программы, «намерения» которой неизвестны.

Стирание массива информации путем записи в него нулевых (или иных) кодов может занять слишком много времени, если, например, объем массива составляет 1М (2^{20}) или более ячеек. Предлагается вместо «реального» проводить «фиктивное» стирание информации. Время выполнения фиктивного стирания массива произ-

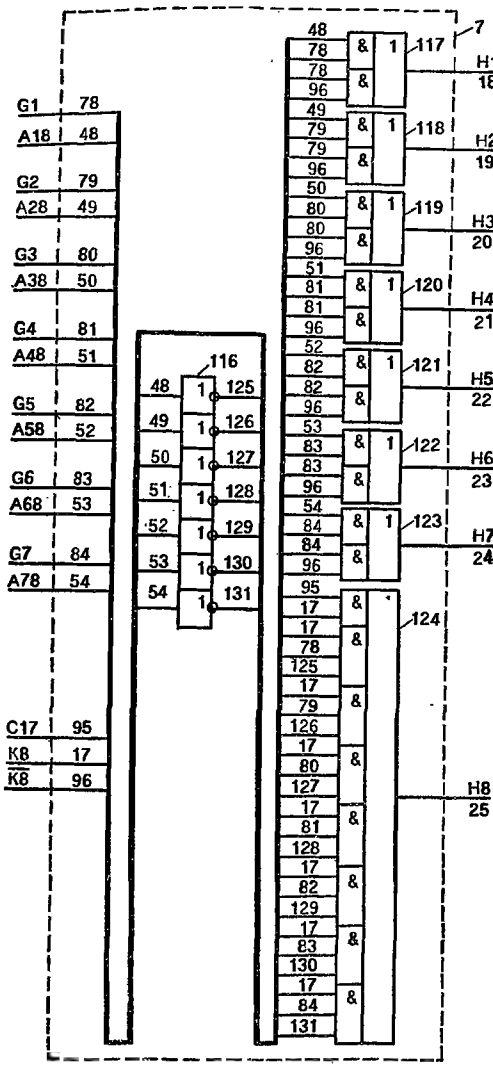


Рис. 7.43. Структура блока 7 (см. рис. 7.38)

вольной длины с произвольно заданными границами соответствует одному — двум обращениям в ОЗУ.

При фиктивном стирании области Z в ОЗУ (рис. 7.44) информация в этой области сохраняется, однако запрещается ее выдача в магистраль. Иными словами, с точки зрения внешнего абонента, в области Z записаны нулевые коды, хотя в действительности это не так. Запись новой информации в область Z разрешена, причем вновь записанные коды могут быть в дальнейшем считаны, а оставшиеся в области Z старые коды по-прежнему оказываются защищенными от считывания.

Рис. 7.44. Процесс фиктивного стирания в ОЗУ старой информации и записи новой

Структура блока памяти с возможностью фиктивного стирания информации приведена на рис. 7.45. В ОЗУ1 хранится полезная информация, в ОЗУ2—ОЗУ7 — признаки, определяющие возможность ее считывания.

Чтобы прочитать информацию D из основного ОЗУ (ОЗУ1) необходимо открыть передающие элементы TR сигналом F=1. Сигнал F=1 формируется только в том случае, когда одновременно из всех вспомогательных ОЗУ (ОЗУ2 — ОЗУ7) через соответствующие мультиплексоры MUX считываются единичные биты. Если хотя бы одно из вспомогательных ОЗУ выдает при считывании сигнал лог. 0, то F=0 и элементы TR остаются закрытыми, т. е. D=0 (ячейка «стерта»).

Каждый бит x_i хранимый в ОЗУ2 (i=0, 1, 2, ..., 1М—1), определяет возможность доступа по чтению к соответствующей 32-разрядной ячейке ОЗУ1. Для того чтобы ячейка была доступна, необходимо (но не достаточно), чтобы $x_i=1$. Если $x_i=0$, то ячейка безусловно считается «стертой».

Каждый бит y_j, хранимый в ОЗУ3 (j=0, 1, 2, ..., 64К—1), определяет истинность соответствующего 16-разрядного слова, хранимого в ОЗУ2. Для того чтобы бит $x_i=1$ и другие единичные биты в ячейке могли служить указателями доступности ячеек ОЗУ1, необходимо (но не достаточно), чтобы $y_j=1$. Если $y_j=0$, то независимо от кода в ячейке ОЗУ2 соответствующие 16 ячеек ОЗУ1 считаются недоступными по чтению («стертыми»).

Аналогично каждый бит в ОЗУ4 может «стереть» 64 ячейки ОЗУ1; каждый бит в ОЗУ5, 6, 7 может «стереть» соответственно 0,5К, 4К и 64К 32-разрядных ячеек ОЗУ1.

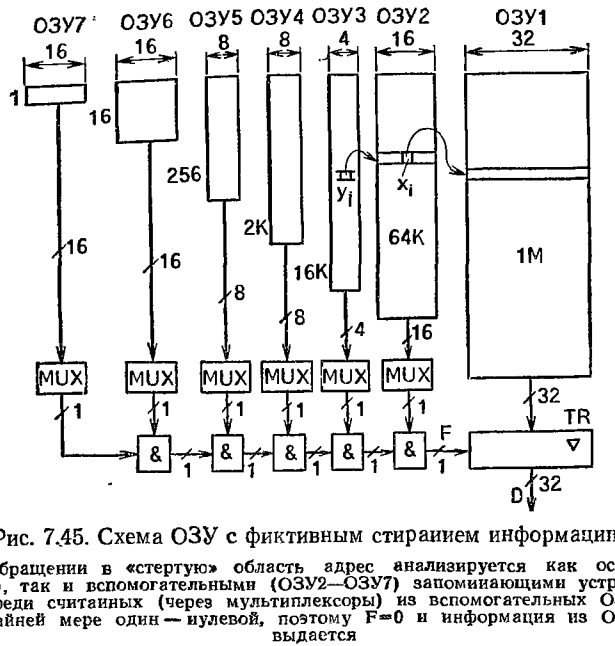


Рис. 7.45. Схема ОЗУ с фиктивным стиранием информации.

При обращении в «стертую» область адрес анализируется как основным (ОЗУ1), так и вспомогательными (ОЗУ2—ОЗУ7) запоминающими устройствами. Среди считанных (через мультиплексоры) из вспомогательных ОЗУ бит по крайней мере один — нулевой, поэтому F=0 и информация из ОЗУ1 не выдается

На рис. 7.46 показана упрощенная (для наглядности) система из четырех ОЗУ. Как и в предыдущем примере, ОЗУ1 — основное, ОЗУ2, 3, 4 — вспомогательные. ОЗУ1 содержит 32 ячейки любой разрядности. Адрес, поступающий в ОЗУ1, содержит 5 разрядов: А4, А3, А2, А1, А0 (А0 — младший разряд). ОЗУ2 содержит 8 4-разрядных ячеек. «Сфера влияния» каждой ячейки ОЗУ2 (ОЗУ3, 4) отмечена на рисунке штриховыми линиями. Ячейки ОЗУ2 адресуются тремя разрядами: А4, А3, А2, биты в ячейке ОЗУ2 адресуются оставшимися двумя разрядами: А1, А0.

ОЗУ3 содержит две 4-разрядные ячейки, адресуемые разрядом А4; биты в ячейках адресуются разрядами А3 и А2. ОЗУ4 содержит одну 2-разрядную ячейку, выбираемую при обращении по любому адресу; биты в этой ячейке выбираются разрядом А4.

Операция фиктивного стирания выполняется схемно, после задания внешним абонентом двух граничных адресов стираемого массива и поступления запускающего сигнала. Чтобы стереть массив информации, лежащий между ячейками 01001 и 10111 (не включая эти ячейки), достаточно записать нули в определенные разряды ячеек ОЗУ2 и ОЗУ3, как показано на рис. 7.46 (см. коды, записанные внутри ячеек). В ОЗУ4 в данном примере информация не меняется, так как и в первой, и во второй половинах ОЗУ1 остались ячейки памяти, доступ к которым по чтению разрешен. «Пустые» разряды ячеек ОЗУ2 и ОЗУ3 хранят старую информацию (0 или 1), которая не представляет интереса при рассмотрении данного при-

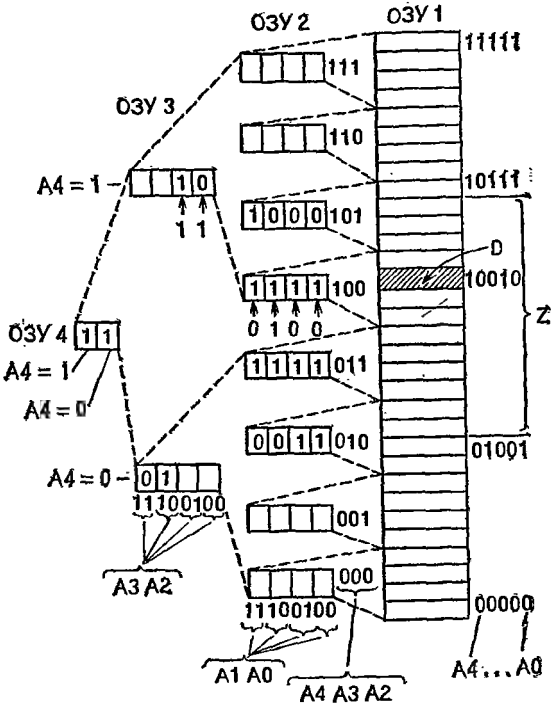


Рис. 7.46. Управление фиктивным стиранием информации.

Для фиктивного стирания области Z в ОЗУ2 и ОЗУ3 записываются нулевые биты (показаны внутри ячеек); при записи кода D в область Z производится коррекция содержимого ячеек ОЗУ2 и ОЗУ3 (новые коды показаны вне ячеек, направления пересылки бит показаны стрелками)

мера. Из рис. 7.46 следует, что в ОЗУ2 нули записаны в две ячейки, соответствующие нижней и верхней границам стираемой области. Это же относится и к ОЗУ3. В общем случае, во всех возможных ситуациях, независимо от объема основного ОЗУ и расположения стираемой области, фиктивное стирание информации сопровождается не более чем двумя параллельными обращениями к группе вспомогательных ОЗУ.

После стирания массива информации в примере, приведенном на рис. 7.46, в ячейку ОЗУ1 с адресом 10010 записывается код D. Чтобы этот код можно было в дальнейшем беспрепятственно считывать, производится коррекция информации в ОЗУ2 и ОЗУ3, как показано на рис. 7.46. Вместо бит, показанных внутри ячеек, записываются биты, показанные вне ячеек (см. соответствующие стрелки, показывающие замену старых бит новыми).

Операции коррекции выполняются с помощью несложных схем.

Так как быстродействие вспомогательных ОЗУ, выполненных на микросхемах статической памяти (или на регистрах), более высокое, чем быстродействие основного блока ОЗУ (рис. 7.45), выполненного на микросхемах динамической памяти, операции коррекции (напоминающие по сути операцию Чтение — модификация — запись) успевают выполниться в течение цикла записи слова данных в основной блок ОЗУ.

7.21. Использование освободившихся при частичной дешифрации разрядов адресной шины для передачи данных [200]

При частичной дешифрации используются не все линии адресной шины (см. гл. 1, 2). Освободившиеся линии можно использовать для передачи данных (а не разрядов адреса), как показано на рис. 7.47.

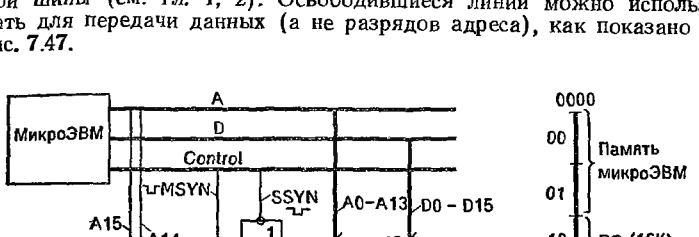


Рис. 7.47. Подключение 30-разрядного выходного порта к 16-разрядной микроЭВМ с возможностью одновременной загрузки всех 30 разрядов:

а — функциональная схема; *б* — распределение адресного пространства

Для загрузки выходного порта микроЭВМ выполняет, например, команду пересылки содержимого регистра общего назначения R4 в ячейку памяти с адресом 10××××××××××××××. Два старших разряда адреса A15 и A14 используются по прямому назначению и служат для выбора 30-разрядного регистра RG, так что этот регистр занимает в адресном пространстве (рис. 7.47, *б*) область размером 16K ячеек. Истинность адреса подтверждается сигналом MSYN (см. описание интерфейса Общая шина, гл. 2). В младших 14 разрядах адреса (A0 — A13) содержатся данные, которые попадают в старшие 14 разрядов выходного порта. Младшие 16 разрядов выходного порта в то же время загружаются обычным путем — с шины данных, на которую выдано содержимое регистра R4. Сигнал с выхода дешифратора воздействует на регистр RG и, пройдя через элемент НЕ, в качестве ответного сигнала SSYN возвращается в процессор. Временна́я диаграмма, сопровож-

дающая загрузку порта, не содержит каких-либо отличий от стандартной диаграммы записи (см. рис. 2.4).

Схема может быть полезной, когда недопустима «постепенная» смена информации на выходах порта, связанная с поэтапной загрузкой регистра, разрядность которого превышает разрядность шины данных микроЭВМ. При поэтапной загрузке на выходах порта в течение относительно большого промежутка времени, например в течение 10 мкс, будет присутствовать промежуточная информация — не старая, но еще и не новая. Это, в свою очередь, может привести к ложному срабатыванию исполнительных устройств, например к появлению ложных светящихся точек на экране электронно-лучевой трубки, если выходной порт содержит координаты точки, сведения о цвете, яркости и т. п.

Таким образом, можно временно «превратить» 16-разрядную микроЭВМ в 30-разрядную.

7.22. Трехпроводная магистраль для передачи переменного напряжения питания, информации и стартовых меток [133]

Магистраль (рис. 7.48) предназначена для обмена информацией между абонентами-источниками и приемниками информации. К магистрали подключаются до 32 источников и до 32 приемников информации. «Магистральное время» складывается из последовательности циклов, а каждый цикл — из 32 периодов переменного напряжения (рис. 7.49). В каждом цикле возможна передача бита от

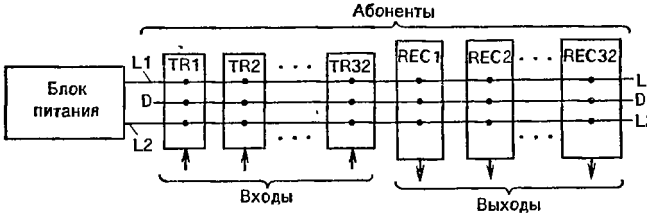


Рис. 7.48. К магистрали подключен блок питания, абоненты-источники TR_i и абоненты-приемники REC_j информации

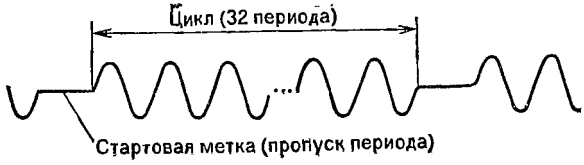


Рис. 7.49. Формат информационного цикла.

В каждом периоде переменного напряжения от одного абонента к другому (от TR_i к REC_j) передается один бит информации (0 или 1)

определенного источника информации к определенному приемнику или группе приемников.

Блок питания обеспечивает абонентов энергией, тактовой частотой и стартовыми метками в виде «пустых» периодов (рис. 7.49). Пустые периоды не влияют на энергоснабжение абонентов, так как у каждого абонента к выходу выпрямителя подключен конденсатор достаточно большой емкости, который отдает ранее накопленную энергию в периоды отсутствия ее поступления из магистрали.

Каждый абонент, обнаружив стартовую метку, начинает подсчет периодов переменного напряжения с помощью счетчика. Когда код в счетчике достигает заданного значения, абонент (а точнее — пара абонентов: передатчик и соответствующий ему приемник) вступает в сеанс связи, при котором из передатчика в приемник поступает один бит информации (1 или 0). Таким образом, в каждом периоде переменного напряжения производится передача одного бита между двумя абонентами, которые активизируются в этот период; остальные абоненты пассивны. (Не исключена возможность передачи бита из одного передатчика одновременно нескольким приемникам.)

Схема, поясняющая принцип взаимодействия абонентов, приведена на рис. 7.50; пример практической реализации ее узлов — на рис. 7.51; временная диаграмма взаимодействия передатчика с приемником — на рис. 7.52.

Если входной сигнал A от объекта, например от контактного датчика, имеет значение лог. 1, то оптронный переключатель SW1 (условно показанный на рис. 7.50 в виде механического переключателя) замыкается при наличии положительной полуволны напряже-

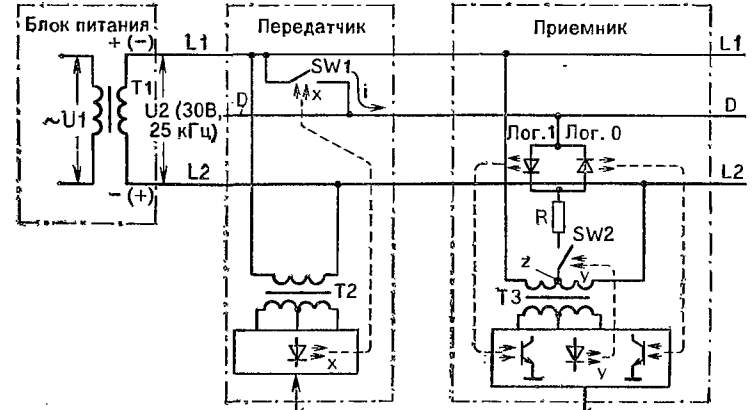


Рис. 7.50. Взаимодействие передатчика с приемником.

Под действием световых потоков «x» и «y» замыкаются ключи SW1 (на полпериода) и SW2 (на целый период). В проводе D течет положительный или отрицательный ток, соответствующий передаче лог. 1 или лог. 0

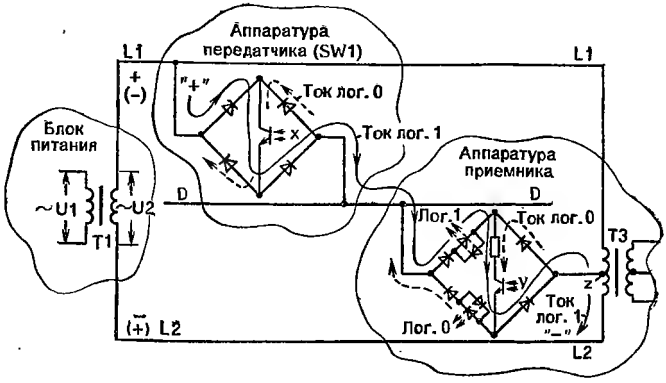


Рис. 7.51. Практическая реализации оптронных переключателей.

Показаны фрагменты схем передатчика и приемника

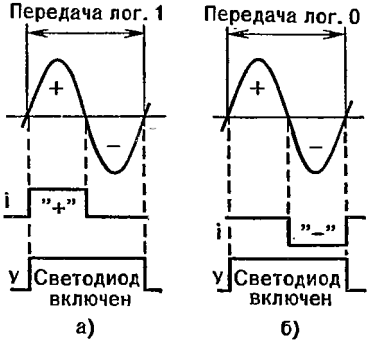


Рис. 7.52. Передача бита от передатчика к приемнику.

Бит передается в течение определенного периода переменного напряжения с использованием избыточного кодирования; лог. 1 и лог. 0 представлены соответственно положительным (*а*) и отрицательным (*б*) токами i в первом и втором полупериодах

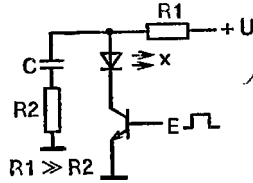


Рис. 7.53. Импульсная схема управления, светодиодом оптрона ключа SW1 (SW2), (см. рис. 7.50).

Повышается надежность системы за счет исключения действия отказавших абонентов на общие цепи

ния заданного периода. Если A=0, то переключатель SW1 замыкается при отрицательной полуволне. Замыкание переключателя SW1 происходит под действием светового потока x от светодиода, управляемого от аппаратуры абонента-источника, которая включает ток через светодиод в нужном интервале времени.

В течение полного периода (положительная и отрицательная полуволны напряжения U2) оптронный переключатель SW2 вы-

бранного приемника замкнут под действием светового потока у. При передаче лог. 1 ток i протекает по цепи (от положительного полюса источника напряжения к отрицательному, рис. 7.50): L1— переключатель SW1 — провод D — светодиод оптрона лог. 1 — резистор R — переключатель SW2 — точка z — провод L2. При передаче лог. 0 ток меняет направление и протекает по цепи: провод L2 — точка z — переключатель SW2 — резистор R — светодиод оптрона лог. 0 — провод D — переключатель SW1 — провод L1. Переданный бит запоминается в приемнике и передается на выход B, например, для управления электродвигателем: при B=0 двигатель выключен, при B=1 — включен.

За счет избыточности представления передаваемого бита (приемник анализирует полярность тока и «своевременность» его протекания) обнаруживаются ошибки передачи, например, вызванные внешними помехами. К таким ошибкам относятся, например, следующие: оба светодиода (лог. 0 и лог. 1) поочередно включались на протяжении одного периода переменного напряжения; ток лог. 1 наблюдался в течение всего периода, а не его первой половины и т. д.

При обнаружении ошибки приемник в зависимости от положения размещенного в нем механического переключателя (на рисунках не показан) либо сохраняет старое значение сигнала B, либо устанавливает B=0, либо устанавливает B=1. Выбор одного из указанных видов реакции на ошибку производится из соображений «здравого смысла» на этапе проектирования системы и зависит от особенностей аппаратуры, подключенной к выходу приемника.

Для дополнительной защиты от сбоев в приемник может быть введена схема сравнения бит, поступивших при двух последовательных обращениях со стороны передатчика в двух сеансах связи, следующих друг за другом. Если новый бит совпадает с принятым при предыдущем обращении, то только в этом случае он передается на выход B. «Неподтвержденный» бит расценивается как ошибочный и игнорируется.

Световые потоки x, y формируются импульсными схемами за счет энергии, накопленной в конденсаторах C (рис. 7.53). При нормальной работе во время действия импульса E светодиод излучает энергию, ранее накопленную в конденсаторе C. При отказе абонента, вероятнее всего, сигнал E не будет меняться, т. е. он постоянно равен 0 или 1. Если E=0, то абонент не мешает соседям обмениваться информацией, конденсатор C заряжен, транзистор закрыт, светодиод не излучает свет. Если сигнал E постоянно равен 1, то конденсатор разряжен и ток через светодиод слишком мал для его нормального свечения. Поэтому отказавший абонент также не мешает соседям обмениваться информацией по магистрали.

7.23. Микропроцессорная управляющая система с битовой и байтовой адресацией ОЗУ [134]

Во многих управляющих микропроцессорных системах программы оперируют битами, отражающими состояние внешних объектов, например датчиков релейного типа. Обработка бит, образующих массивы, ведется по правилам булевой алгебры. Для уменьшения объема памяти программ и данных и повышения производительности

системы предлагается использовать ОЗУ с возможностью битовой и байтовой адресации.

При байтовой адресации памяти, содержащей 512 8-разрядных ячеек, из центрального процессора в адресную шину магистрали поступает 16-разрядный адрес (рис. 7.54, *а*), в котором в трех старших разрядах A15—A13 содержатся 0. Разряд A12 может иметь значение 0 или 1 (не используется). Адрес одной из 512 8-разрядных ячеек содержится в разрядах A11—A3, младшие разряды A2—A0 не используются.

При битовой адресации (рис. 7.54, б) A15=A14=0, A13=1. Если A12=0, то бит записывается или считывается в прямом коде, если A12=1, то в обратном. Позиция бита в 8-разрядном слове, выбираемом из ОЗУ в соответствии с разрядами A11—A3 адреса, определяется разрядами A2—A0.

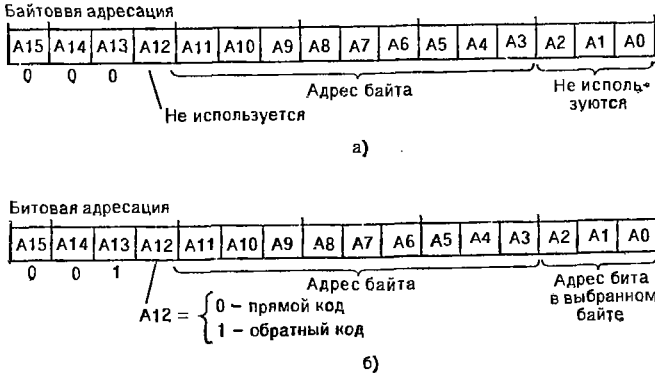


Рис. 7.54. Формат адреса, выдаваемого процессором при обращении к ОЗУ:

а — при адресации к байту; *б* — при адресации к биту в выбранном байте

Предположим, что в ячейке 375_8 ОЗУ записан 8-разрядный код $10111010_2=272_8$ (рис. 7.55). Для считывания этого кода в режиме байтовой адресации процессор формирует 16-разрядный адрес 003750_8. Если процессору предписано извлечь из этой ячейки бит D4 в обратном коде, то он использует битовую адресацию и формирует адрес 033754. В результате считаный бит инвертируется и передается по определенной линии шины данных магистрали в процессор, в данном примере — по линии D7.

Оперативное запоминающее устройство (рис. 7.56) выполнено на 8 БИС статической памяти со структурой 512×1 бит. Входы CE, WE и DI служат соответственно для «выбора кристалла», разрешения записи и задания входного бита; на выход DO БИС поступает считанный бит. Усилители TR1—TR3 транслируют входные сигналы на выходы только при E1=E2=0, в противном случае усилители находятся в пассивном (выключенном) состоянии. Мультиплексор MUX транслирует на прямой и инверсный выходы сигнал с входа, выбран-

ного разрядами A0—A2 адреса, дешифратор DC срабатывает при E1=E2=0, E3=1. Элемент суммирования по модулю два (M2) в зависимости от значения сигнала A12 транслирует сигнал D7 в прямом или обратном коде.

При обращении к ОЗУ процессор формирует адрес, данные (при записи), сигналы $\overline{RAM}$=L (напряжение низкого уровня), R/$\overline{W}$, равный L или H (запись, чтение), а также синхросигналы E и Q (H-уровни). При битовой адресации дешифратор DC выбирает нужную микросхему ОЗУ, при байтовой адресации все микросхемы ОЗУ работают синхронно.

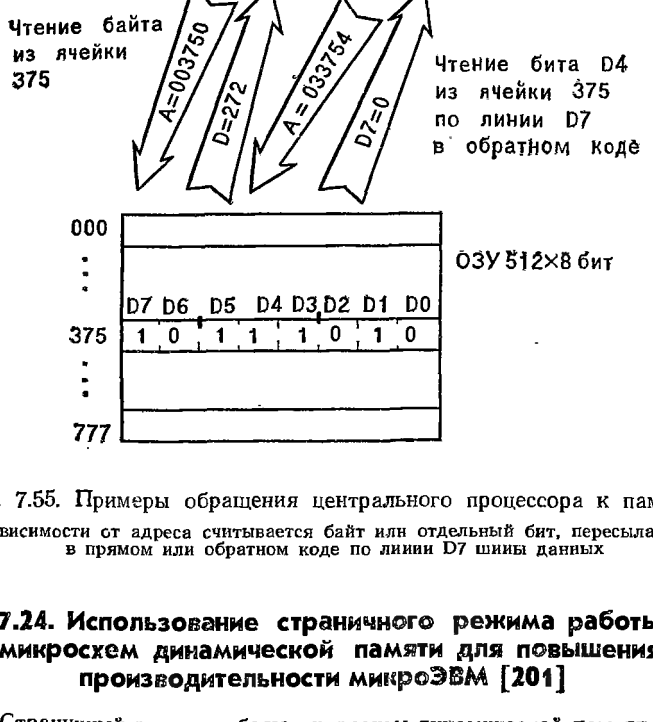


Рис. 7.55. Примеры обращения центрального процессора к памяти.

В зависимости от адреса считывается байт или отдельный бит, пересылаемый в прямом или обратном коде по линии D7 шины данных

7.24. Использование страничного режима работы микросхем динамической памяти для повышения производительности микроЭВМ [201]

Страничный режим работы микросхем динамической памяти позволяет примерно в два раза уменьшать время доступа к ячейке ОЗУ. Однако этот режим может быть использован при многократном обращении активного устройства к одной и той же странице памяти. Если активным устройством является процессор, то он обычно работает не с одной, а сразу с несколькими страницами, соответствующими областям программы и данных, причем обращения к этим

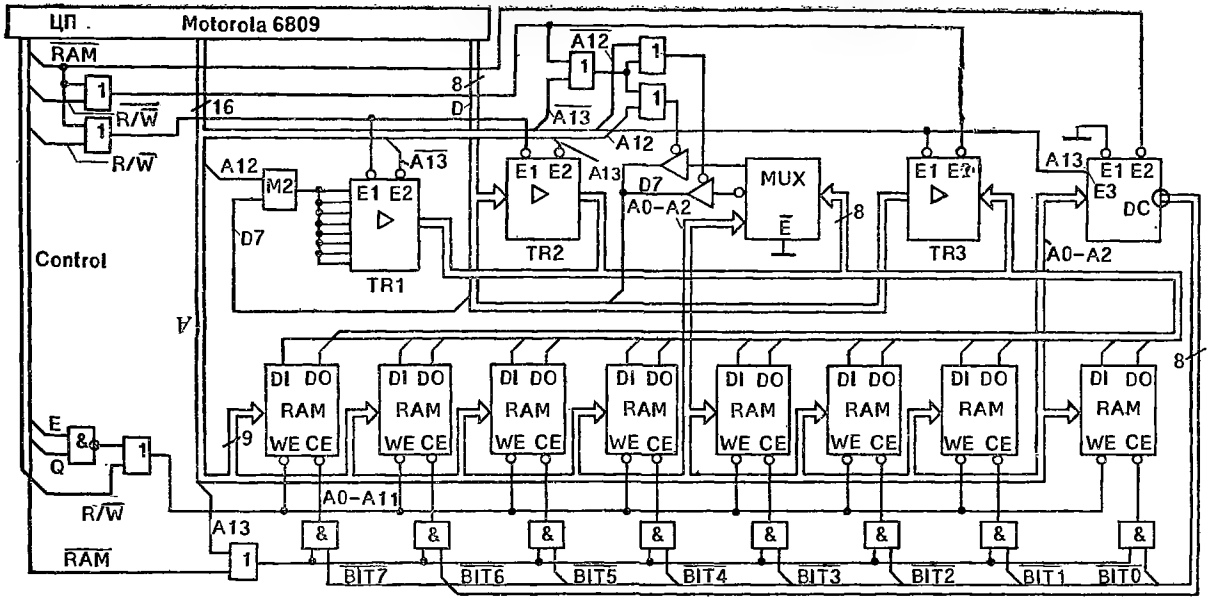


Рис. 7.56. Микропроцессорная система с возможностью битовой и байтовой адресации ОЗУ

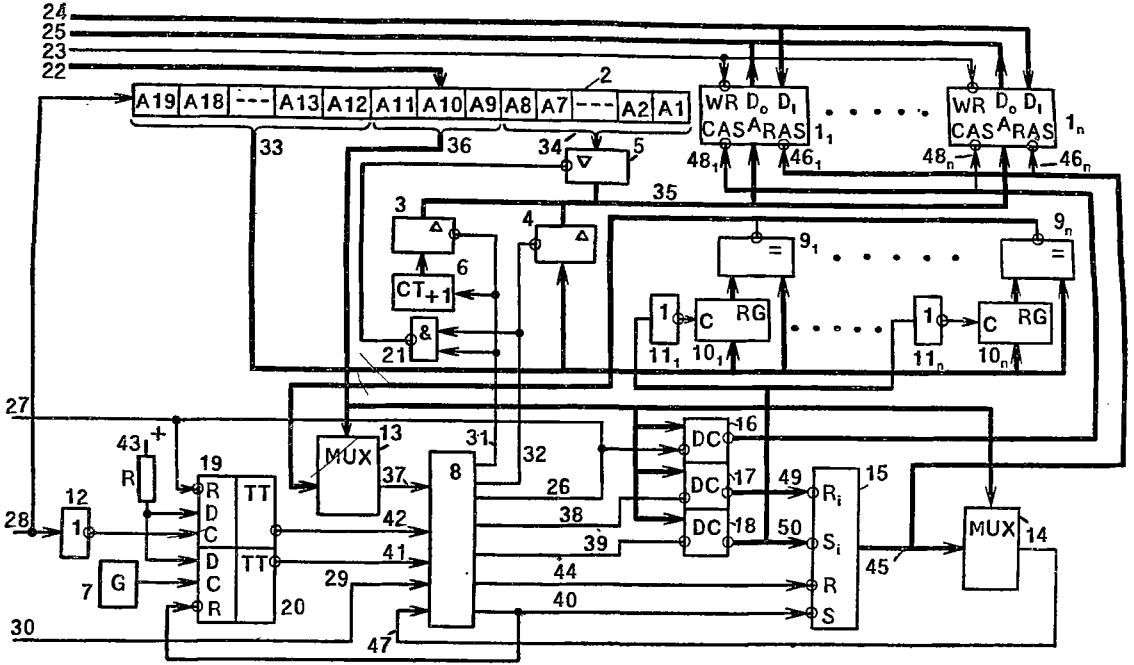


Рис. 7.57. Функциональная схема ОЗУ с ускоренным доступом к ячейкам памяти за счет использования страничного режима работы микросхем

страницам в основном чередуются. Поэтому на первый взгляд может показаться, что странчный режим работы микросхем памяти не может обеспечнть ускорение работы мнкроЭВМ.

В предлагаемом схемном решении ОЗУ учитывается отмеченное «непостоянство» процессора при выборе страницы и в то же время обеспечивается ускоренный доступ к ячейкам памяти за счет использования странчного режима работы БИС.

Оперативное запоминающее устройство (рис. 7.57) содержит n накопительных блоков 1_1—1_n (n далее принято равным 8), регистр адреса 2, блоки 3—5 передающих элементов, двоичный счетчнк 6, генератор импульсов 7, блок управления 8, а также n компараторов 9_1—9_n, n регистров старших разрядов адреса 10_1—10_n, группу из n элементов НЕ 11_1—11_n, элемент НЕ 12, мультиплексоры 13, 14, регистр 15 сигналов выбора строки, дешифраторы 16—18, триггеры 19, 20 и элемент И—НЕ 21.

Информационные входы регистра адреса 2 соединены с адресными входами 22 устройства, входы записи накопительных блоков 1_1—1_n — с входом 23 записи устройства, входы и выходы данных накопительных блоков 1_1—1_n — с входами 24 и выходами 25 данных устройства, выход 26 блока управления 8 — с выходом 27 ответа устройства, управляющнй вход регнстра адреса 2 — с входом 28 запуска устройства, вход 29 блока управления 8 — с входом 30 начальной установки устройства.

Выход 31 блока управлення 8 соединен с управляющими входами блока 3 и счетчнка 6, выход 32 — с управляющим входом блока 4, информационные входы которого соединены с первой группой 33 выходов регистра адреса 2, иинформационные входы блока 5 соединены с группой 34 выходов регистра адреса 2. Информационные входы блока 3 соединены с выходами двоичного счетчика 6. Выходы блоков 3, 4 и 5 соединены с адресными входами 35 накопительных блоков 1_1—1_n.

Управляющие входы мультиплексоров 13 и 14 соединены с информационными входами дешифраторов 16—18 и с группой 36 выходов регистра адреса 2. Выходы n компараторов 9_1—9_n соединены с информацнонными входами мультиплексора 13, выход которого соединен с входом 37 блока управления 8. Выходы регистров 10_1—10_n старших разрядов адреса соединены с первыми группами входов соответствующих компараторов 9_1—9_n, вторые группы входов которых соединены с информацнонными входами регистров 10_1—10_n и с группой 33 выходов регистра 2.

Управляющие входы дешифраторов 16—18 соединены соответственно с выходами 26, 38 и 39 блока управления 8. Вход установки в 0 триггера 19 соединен с выходом 26 блока 8, вход установки в 0 триггера 20 соединен с входом общей установки единицы регистра 15 и с выходом 40 блока 8.

Выход генератора импульсов 7 соединен с входом синхронизации триггера 20, нулевой выход которого соеднпен с входом 41 блока 8. Управляющий вход регистра адреса 2 через элемент НЕ 12 соединен с входом синхронизации триггера 19, нулевой выход которого соединен с входом 42 блока 8. Информационные входы триггеров 19 и 20 соединены через резистор с шиной 43 положительного источника питания устройства.

Выход 44 блока управления 8 соединен с входом общей установки в 0 регистра 15, выходы 45 которого соединены с информацнонными входами мультиплексора 14 и с индивидуальными входами

46_1—46_n выбора строки n накопительных блоков 1_1—1_n. Выход мультиплексора 14 соединиеи с входом 47 блока управления 8.

Выходы дешифратора 16 соединены с иидивидуальными входами 48_1—48_n выбора столбца n накопительных блоков 1_1—1_n. Выходы дешифратора 17 соединеиы с разрядными входами 49 устаиовки в 0 регистра 15. Выходы дешифратора 18 соединены с разрядными входами 50 установки 1 регистра 15 и с входами элементов НЕ 11_1—11_n, выходы которых соединены с управляющими входами регистров 10_1—10_n.

Накопительный блок 1_i (i=1, 2, ..., n), показанный на рис. 7.58, содержит m БИС запомииающих устройств 51_1—51_m (m — разрядность слов, хранимых в ОЗУ). БИС 51_j (j=1, 2, ..., m) содержит группу адресиых входов 52, соединиеиных с группой адресиых входов

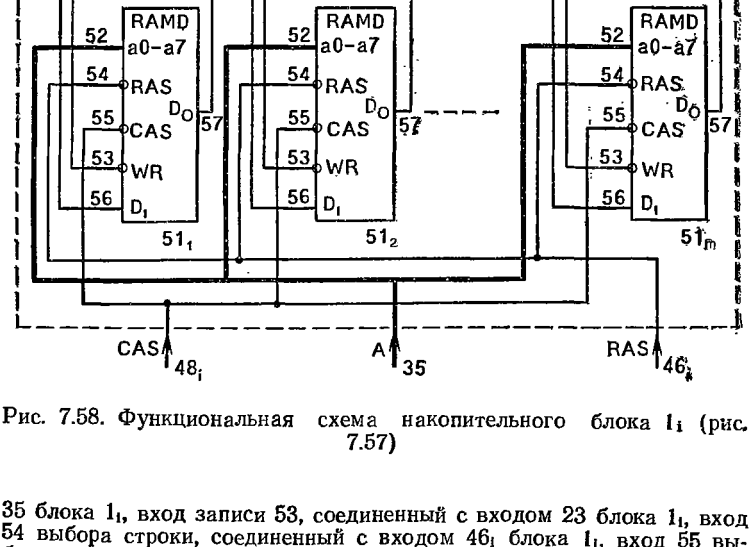


Рис. 7.58. Функциональная схема накопительного блока 1_i (рис. 7.57)

35 блока 1_i, вход записи 53, соединенный с входом 23 блока 1_i, вход 54 выбора строки, соединенный с входом 46_i блока 1_i, вход 55 выбора столбца, соединенный с входом 48_i блока 1_i, вход 56 данных, входящий в состав группы входов 24 блока 1_i, выход 57 данных, входящий в состав группы выходов 25 блока 1_i.

Регистр 15 сигналов выбора строки в приведенном на рис. 7.59 примере (n=8) содержит элементы И 58, 59 и регистр 60, состоящий из триггеров 61. Входы 50 регистра 15 соединены с первыми входами элементов И 58, вторые входы которых соединиены с входом 40 регистра 15. Входы 49 регистра 15 соединены с первыми входами элементов И 59, вторые входы которых соединены с входом 44 ре-

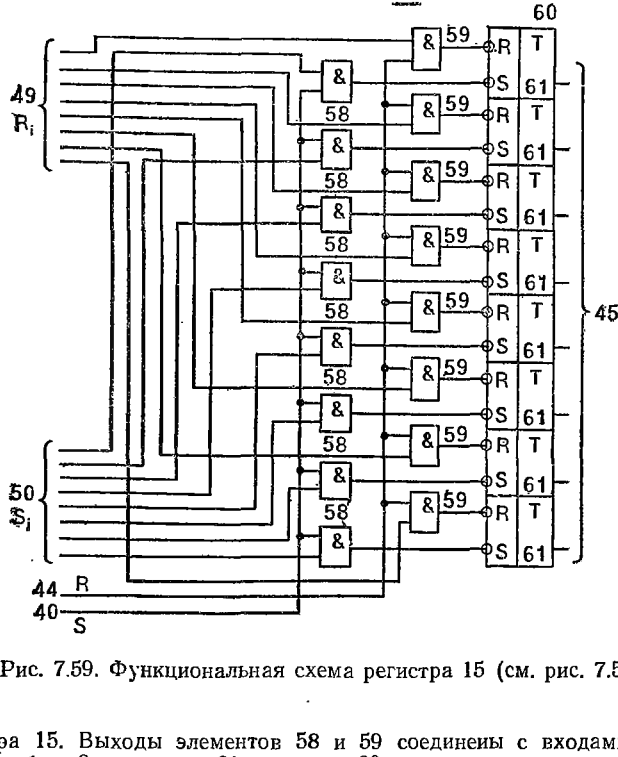


Рис. 7.59. Функциональная схема регистра 15 (см. рис. 7.57)

гистра 15. Выходы элементов 58 и 59 соединеиы с входами установки 1 и 0 триггеров 61 регистра 60, выходы которого являются выходами 45 регистра 15.

Блок управления 8 (рис. 7.60) содержит генератор импульсов 62, входной регистр 63, выходной регистр 64, элементы И 65, 66, элемент НЕ 67 и логический блок 68. Вход 29 блока 8 соединен с входами установки 1 регистров 63, 64 и с входами элементов И 65, 66. Выход генератора импульсов 62 соединен с входом синхронизации регистра 63 и с входом элемента НЕ 67, выход которого соединен с входом синхронизации регистра 64.

Блок 68 может быть выполнен в виде ПЗУ, ПЛМ или комбинационной схемы из стандартных логических элементов, как показано в примере, приведенном на рис. 7.61, где блок 68 содержит элементы НЕ 96—100, элементы И—НЕ 101—104, элементы И 105—112 и дешифратор 113.

Рассмотрим работу составных частей устройства.

Накопительные блоки 1_1—1_n (рис. 7.57) выполнены на БИС 51_1—51_m (рис. 7.58) запоминающих устройств динамического типа, в данном примере — на БИС типа К565РУ5. БИС 51_j имеет восемь адресных входов a0—a7, вход RAS выбора строки, вход CAS выбора

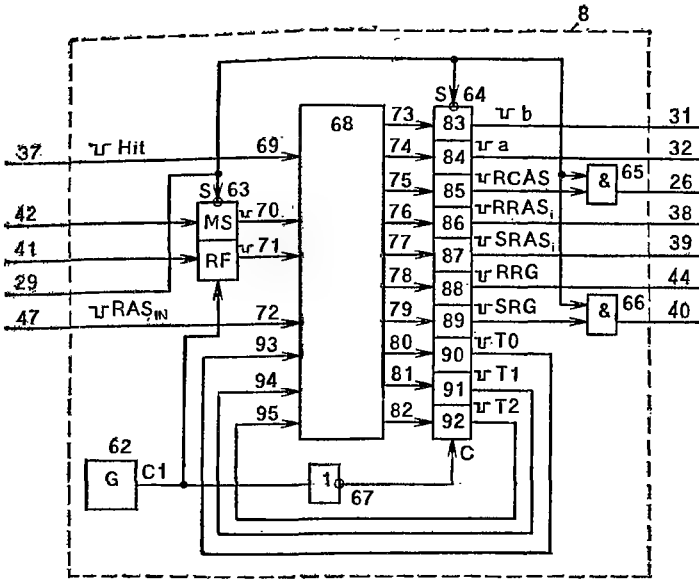


Рис. 7.60. Функциональная схема блока управления 8 (см. рис. 7.57)

столбца, вход WR записи, вход D_I и выход D_O данных. Информационная емкость БИС 51_J составляет 64К бит (К=1024).

Работа БИС 51_J в режиме записи поясняется временными диаграммами, приведенными на рис. 7.62, *а*. В момент формирования отрицательного фронта сигнала RAS (при переходе напряжения высокого уровня H в напряжение низкого уровня L) на адресных входах A (a0—a7) установлен некоторый 8-разрядный код, который запоминается во внутреннем регистре БИС 51_J и используется для выбора «строки» внутреннего накопителя (точнее говоря, используются только семь разрядов, а восьмой используется в дальнейшем, при выборе «столбца»). Аналогично, по отрицательному фронту сигнала CAS с адресных входов A считывается другая часть полного 16-разрядного адреса, которая совместно с принятым ранее разрядом рассматривается как адрес столбца внутреннего накопителя. Накопитель выполнен в виде матрицы, содержащей 128 строк и 512 столбцов. По отрицательному фронту сигнала CAS с линии WR считывается сигнал записи (уровень L), а с линии входных данных D_I — уровень L или H в зависимости от значения записываемого бита.

Далее БИС 51_J производит запись поступившего бита в нужную ячейку памяти (при этом источник сигналов RAS и CAS должен выждать некоторое время), после чего сигналы RAS и CAS снимаются, т. е. переходят в состояние H. Заштрихованные участки временных диаграмм соответствуют произвольным значениям сигналов.

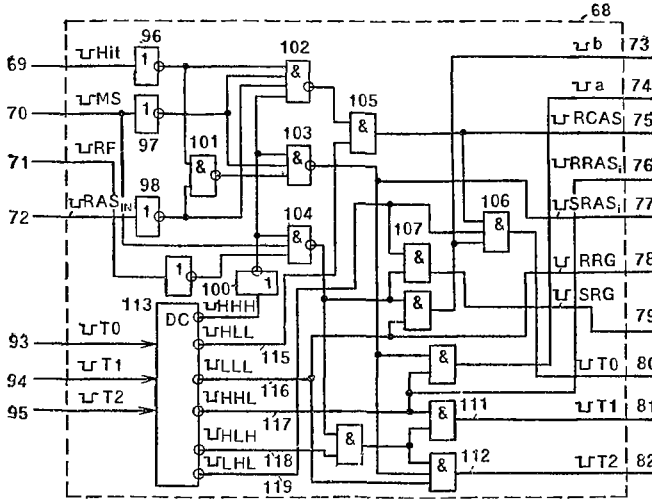


Рис. 7.61. Функциональная схема логического блока 68 (см. рис. 7.60)

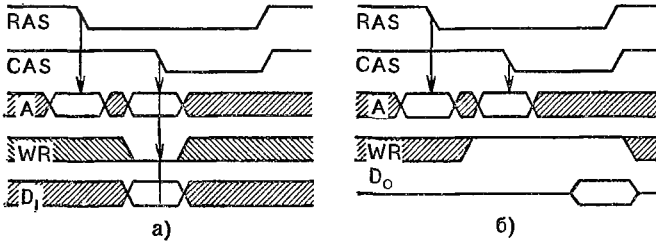


Рис. 7.62. Временные диаграммы записи *(а)* и чтения *(б)* информации в обычном режиме

При считывании бита информации из некоторой ячейки БИС 51$_i$ (см. рис. 7.62, *б*) адрес также передается в два этапа — по отрицательным фронтам сигналов RAS и CAS, причем по отрицательному фронту сигнала CAS с линии WR считывается уровень H (признак чтения). Через некоторое время на выходной линии D_O устаиавливается напряжение высокого или низкого уровня, в зависимости от значения считанного бита. Цикл считывания завершается снятием сигналов RAS и CAS, после чего выход D_O переходит в исходное состояние с высоким сопротивлением.

Работа устройства основана на использовании режимов стра-

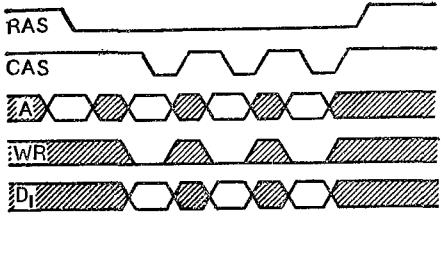


Рис. 7.63. Временны́е диаграммы записи информации в страничном режиме

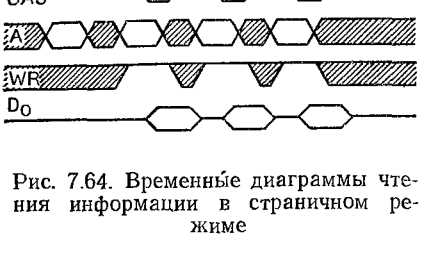


Рис. 7.64. Временны́е диаграммы чтения информации в страничном режиме

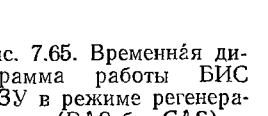


Рис. 7.65. Временна́я диаграмма работы БИС ОЗУ в режиме регенерации (RAS без CAS)

ничной записи (рис. 7.63) и страничного чтения (ри́с. 7.64). Первое обращение к БИС 51_J, как и в обычных режимах записи и считывания (см. рис. 7.62), сопровождается выдачей уровней L на входы RAS и CAS. Однако в дальнейшем сигнал RAS в течение некоторого времени остается в состоянии L, а сигнал CAS меняет состояние. На рис. 7.63, 7.64 сигнал CAS принимает значение L три раза, что соответствует трем обращениям к БИС 51_J. При каждом обращении полный 16-разрядный адрес складывается из двух частей: первая часть (старшие 8 разрядов) передается в БИС 51_J по отрицательному фронту сигнала RAS и сохраняется во внутреннем регистре адреса; вторая часть (младшие 8 разрядов) принимается с адресных входов A по отрицательным фронтам сигнала CAS и каждый раз «подстыковывается» к неизменным старшим разрядам.

Таким образом, в пределах фиксированной «страницы» с номером, определяемым старшими 8 разрядами адреса, можно в быстром темпе, в произвольной последовательности выбирать ячейки памяти, определяемые младшими 8 разрядами адреса. Иными словами, для обращения к ячейкам памяти, лежащим в пределах одной и той же страницы, не нужно всякий раз формировать пару сигналов RAS—CAS (см. рис. 7.62); достаточно один раз указать номер страницы (рис. 7.63, 7.64), а затем, меняя в нужной последовательности младшие разряды адреса и оперируя сигналом CAS (WR, D_I, D_O), выбирать нужные ячейки памяти в пределах страницы.

Отметим, что режим страничной записи и считывания, так же как и другие режимы, не исключает необходимости регенерации данных. Регенерация всей матрицы (128 строк) должна проводиться

с периодом не более 2 мс (для данного типа БИС). Регенерация строк должна проводиться с темпом 2/128 мс ≈ 15 мкс (не реже). В каждом цикле регенерации (рис. 7.65) на адресные входы A подается очередной код с 7-разрядного двоичного счетчика (при переполнении счет начинается с нуля) и сигнал RAS, по отрицательному фронту которого код принимается в БИС и используется в качестве номера строки, которая подвергается регенерации.

В накопительном блоке 1_i в каждой БИС 51_j хранится 64К бит (один разряд m-разрядного слова данных или команды); в целом накопительный блок 1_i хранит 64К m-разрядных слов. При обращении к блоку 1_i на все БИС 51_j поступают одинаковые сочетания управляющих и адресных сигналов. Данные с входов 24 в режиме записи поступают на входы D_I БИС и запоминаются. При считывании данные с выходов D_O всех БИС поступают на выходы 25. Блок 1_i, так же как и его составные части (БИС 51_j), работает в режиме записи, считывания, страничной записи и считывания, регенерации (см. рис. 7.62—7.65).

Накопительные блоки 1_1—1_n имеют общую информационную емкость, равную n×64К m-разрядных слов.

Регистр адреса 2 при наличии напряжения высокого уровня (H) на управляющем входе 28 транслирует сигналы с входов 22 на выходы 33, 34, 36 без их запоминания, что позволяет уменьшить время установления правильного сигнала в точке 37 по сравнению с использованием регистра с импульсным управляющим входом. При переходе сигнала на входе 28 в состояние L текущая информация запоминается в регистре 2.

Блок передающих элементов 3 при наличии H-уровня на управляющем входе 31 выключен, т. е. все передающие элементы находятся в состоянии с высоким выходным сопротивлением. При наличии L-уровня на управляющем входе 31 блок передающих элементов 3 переходит в активное состояние и транслирует информацию с входов на выходы без изменения фазы. Блоки 4 и 5 работают аналогично.

Двоичный счетчик 6 формирует адрес очередной строки накопителей, размещенных в БИС 51_j, когда выполняется режим регенерации (рис. 7.65). Исходное состояние счетчика 6 может быть любым; прибавление единицы к его содержимому производится по положительному фронту сигнала на управляющем входе 31.

Генератор импульсов 7 вырабатывает непрерывную последовательность импульсов с периодом повторения (15 мкс), равным периоду времени, отведенному для регенерации очередной строки накопителей. Блок управления 8 анализирует входные сигналы и в зависимости от их сочетания формирует на выходах последовательности управляющих импульсов, необходимые для выполнения одного из трех возможных режимов: сокращенного цикла, полного цикла и регенерации данных (рис. 7.66—7.68). Подробное описание работы блока управления 8 будет приведено после описания принципа действия устройства в целом.

Компаратор 9_i вырабатывает на выходе напряжение низкого уровня (L) при совпадении двух 8-разрядных кодов, поступающих на его входы. При несовпадении кодов на выходе компаратора формируется H-уровень.

Регистр 10_i выполнен аналогично регистру 2 — с потенциальным управляющим входом. (Можно использовать регистр с импульсным

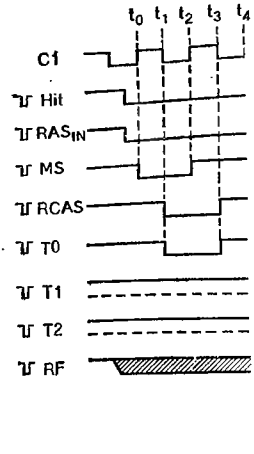


Рис. 7.66. Временные диаграммы работы блока управления 8 (см. рис. 7.57) в режиме сокращенного цикла

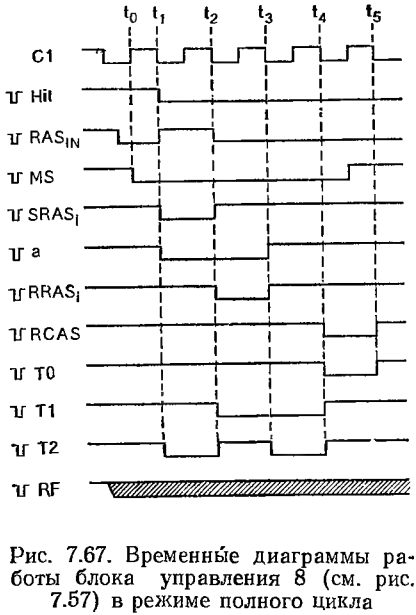


Рис. 7.67. Временные диаграммы работы блока управления 8 (см. рис. 7.57) в режиме полного цикла

управляющим входом, однако он имеет более сложную внутреннюю структуру.)

Мультиплексор 13 в зависимости от кода на управляющих входах 36 транслирует один из входных информационных сигналов, поступающих с выходов компараторов 9_1—9_n, на выход 37. Например, если на всех входах 36 присутствуют сигналы низкого уровня (L), то на выход 37 передается сигнал с выхода компаратора 9_1; если в разрядах A11, A10, A9 регистра адреса 2 присутствует комбинация LLH, то на выход 37 передается сигнал с выхода компаратора 9_2 и т. д. Мультиплексор 14 работает аналогично.

Регистр 15 хранит информацию при наличии H-уровней на всех его входах. При воздействии уровня L на вход 44 производится общий сброс регистра, а при воздействии уровня L на вход 40 — общая установка всех его разрядов в единичное состояние (H). При воздействии уровня L на один из входов групп 49 или 50 производится установка нуля или единицы в соответствующих триггерах 61 регистра 60 (см. рис. 7.59).

Дешифратор 16 при наличии H-уровня на управляющем входе 26 формирует H-уровни на всех своих выходах. При наличии разрешающего сигнала (L-уровня) на управляющем входе 26 дешифратор 16 анализирует код, поступающий на его информационные входы с выходов 36 регистра 2, и формирует L-уровень на одном из своих

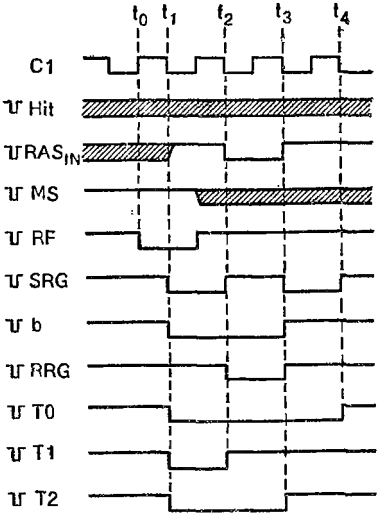


Рис. 7.68. Временные диаграммы работы блока управления 8 (см. рис. 7.57) в режиме регенерации

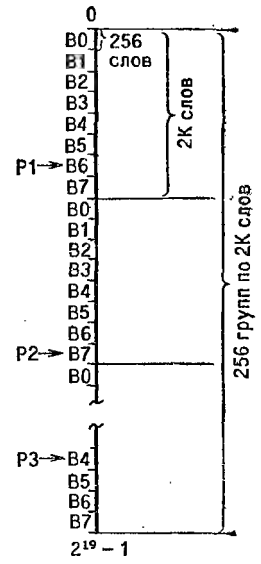


Рис. 7.69. Распределение адресного пространства

Каждый блок 1_1—1_8 хранит 256 разрозненных групп по 256 слов в каждой группе, что создает предпосылки для эффективного использования страничного режима

выходов. Если в разрядах A11, A10, A9 регистра 2 присутствует комбинация сигналов LLL, то L-уровень формируется на линии 48_1. При комбинации LLH L-уровень формируется на линии 48_2 и т. д. Дешифраторы 17 и 18 работают аналогично.

Триггер 19 срабатывает при поступлении на его вход C положительного фронта сигнала. При этом лог. 1, представленная положительным напряжением высокого уровня, записывается в триггер и в виде L-уровня с нулевого плеча триггера поступает на вход 42 блока 8. Сброс триггера 19 производится при подаче L-уровня на линию 26. Триггер 20 работает аналогично.

Далее рассмотрена работа предлагаемого устройства в составе микроЭВМ.

Работа процессора микроЭВМ с памятью характеризуется тем, что адреса ячеек, к которым обращается процессор, не являются «случайными», а группируются в некоторые множества, относительно стабильные во времени. Это связано с тем, что программы имеют циклический характер, а данные, которыми оперирует процессор, чаще всего не «разбросаны» по памяти, а сосредоточены в виде одного или нескольких массивов. Предположим, что области памяти

P1, P2 и P3 (рис. 7.69) представляют собой область программ, область «локальных» данных и область «глобальных» данных, которыми оперирует процессор в течение относительно длительного промежутка времени, например в течение 200 мкс.

Адрес, поступающий из процессора (19 разрядов), запоминается в регистре 2 (см. рис. 7.57). Старшие 8 разрядов адреса A12—A19 «делят» адресное пространство ОЗУ на $2^8=256$ групп, каждая из которых содержит $2^{19-8}=2^{11}=2$К слов, как показано на рис. 7.69. В свою очередь, каждая группа из 2К слов «делится» тремя разрядами адреса A11, A10, A9 на $2^3=8$ зон B0—B7 по 256 слов в каждой. Существено, что каждая зона физически размещена в «своем» накопительном блоке из множества блоков 1_1—1_8.

Адреса 0—255_{10} соответствуют зоне B0 и накопительному блоку 1_1; адреса 256—511_{10} — зоне B1 и блоку 1_2 и т. д. Адрес 2047_{10} соответствует зоне B7 и блоку 1_8, а адрес 2048_{10} — зоне B0 и блоку 1_1, как и нулевой адрес. Иными словами, блок 1_1 хранит не сплошной массив ячеек ОЗУ с последовательно возрастающими адресами, а 256 разрозненных фрагментов массива — 256 зон B0, показанных на рис. 7.69. То же относится и к блокам 1_2—1_8. Отметим, что «с точки зрения» блока 1_1 никакой разрозненности нет — блок хранит 64К m-разрядных слов и «не знает» о своей роли в распределении адресного пространства в целом. Приведенное на рис. 7.69 распределение адресного пространства определяется выбором разрядов, входящих в группы 33, 34 и 36. При ином выборе разрядов можно получить иное распределение адресного пространства.

Области P1, P2, P3 размещены в адресном пространстве случайным образом. Точнее говоря, их положение в адресном пространстве предопределено программным обеспечением микроЭВМ и зависит от времени; однако в развитых системах программного обеспечения заранее предсказать положение этих областей практически невозможно.

На рис. 7.69 показана одна из возможных (благоприятных) ситуаций, когда область P1 попадает в зону B6, область P2 — в зону B7 и область P3 — в зону B4. Другими словами, текущий фрагмент программы размещен в блоке 1_7, а данные, которыми она оперирует, — в блоках 1_8 и 1_5. Такое или иное «бескоифликтное» попадание областей P1, P2 и P3 в блоки 1_1—1_8 обеспечивает максимальный выигрыш в быстродействии. С другой стороны, выигрыш полностью отсутствует, если области P1, P2 и P3 попадают в один и тот же блок из ряда 1_1—1_8. Однако вероятность такого попадания невелика. Действительно, вероятность одновременного попадания областей P1, P2, P3 в блок 1_1 составляет

$$\frac{1}{8}\cdot\frac{1}{8}\cdot\frac{1}{8}=\frac{1}{512}.$$

С учетом наличия восьми равноправных блоков 1_1—1_8 вероятность одновременного попадания областей P1, P2 и P3 в один (любой) из блоков ряда 1_1—1_8 в восемь раз большая:

$$8\cdot\frac{1}{512}=\frac{1}{64}.$$

При наличии конфликтов между двумя областями из трех быстродействие теряется только при строго попеременных обращениях процессора к конкурирующим областям.

При бесконфликтном отображении областей P1, P2 и P3 на блоки 1_1—1_8 оказывается возможным использовать одиовременно в трех соответствующих блоках режимы страиичной записи и считывания, описанные ранее (см. рис. 7.63, 7.64), при которых время доступа к ячейкам памяти сокращается.

После завершеиия очередного цикла регенерации (см. рис. 7.65) устройство переходит в исходное состояиие, при котором сигналы RAS, CAS, поступающие на все БИС 51_j всех блоков 1_i, принимают пассивное значеиие (H).

При первом (после регенерации) обращении процессора в область P1 (P2, P3) на входах блока 1_7 (1_8, 1_5) формируется пара сигналов RAS—CAS, причем сигнал RAS не сиимается, а номер страницы (разряды A12—A19, группа 33) запомииается в регистре 10_7 (10_8, 10_5). Иными словами, при первом обращении реализуется начальный участок временнóй диаграммы, приведениой на рис. 7.63 или 7.64, с запоминанием текущего номера страиицы. При последующем обращении процессора в область P1 (P2, P3) компаратор 9_7 (9_8, 9_5) регистрирует совпадение поступившего номера страницы с ранее использоваииым. Это означает, что создались условия для продолжеиия иачатого ранее страничного режима работы выбранного накопительного блока (см. рис. 7.63, 7.64). Поэтому с помощью блока управления 8 и дешифратора 16 вырабатывается импульсный сигнал CAS, поступающий в блок 1_7 (1_8, 1_5), а сигнал RAS, поступающий в блок 1_7 (1_8, 1_5), по-прежнему остается в активном состоянии (L).

В течение промежутка временн между циклами регенерации (15 мкс) процессор многократно, в нужной ему последовательности, обращается к областям программы и данных (P1, P2 и P3); при этом в блоках 1_7, 1_8 и 1_5 реализуются режимы страннчной записи и считывания.

Если номер текущей страницы при обращении к блоку 1_7 (1_8, 1_5) не совпадает с ранее принятым в регистр 10_7 (10_8, 10_5), то блок управления 8 формирует последовательность сигналов, обеспечивающих снятие сигнала RAS с входа 46_7 (46_8, 46_5) с последующей отработкой процедуры первого обращения к ячейке памяти в страничном режиме (см. начальные участки временных диаграмм, приведенных на рис. 7.63 и 7.64). При этом номер новой страницы запоминается в регистре 10_7 (10_8, 10_5) в расчете на то, что в дальнейшем можно будет продолжить работу в страинчном режиме. При обработке очередного запроса на регенерацию устройство возвращается в пассивное состояние, при котором, в частности, ранее «накопленные» в регистре 15 сигналы RAS переходят в состояиие H.

Далее рассмотрена работа устройства в более подробном виде. Устройство приводится в исходное состояние при поступлении L-уровня на вход 30 начальной установки (см. рис. 7.57); при этом на входы 23 и 28 поданы H-уровни. Сигнал с входа 30 устройства поступает на вход 29 блока управления 8 и далее (см. рис. 7.60) устанавливает в единичное состояние регистры 63 и 64, на выходах которых устанавливаются H-уровни. В то же время сигнал сброса (L) воздействует на входы элементов И 65 и 66 и передается на их выходы, что приводит к сбросу триггеров 19, 20 (на их нулевых выходах устанавливаются H-уровни) и к установке всех разрядов регистра 15 в единичные состояния (H-уровни на выходах). На время действия сигнала сброса открывается дешифратор 16 и иа один из блоков 1_1—1_8 поступает сигнал CAS, что, однако, не может привести к искажению хранимой в блоке информации, так как на входе

23 (WR) присутствует Н-уровень. Начальная установка устройства завершается снятием L-уровня с входа 30.

Устройство работает в трех режимах: сокращенного цикла, полного цикла и регенерации.

Режим сокращенного цикла. Этот режим соответствует рассмотренной ранее благоприятной ситуации, когда текущий номер страницы совпадает с предыдущим, что позволяет продолжить отработку ранее начатой процедуры страничного обращения (см. рис. 7.63, 7.64).

При обращении к ОЗУ на адресных входах 22 устройства устанавливается 19-разрядный код адреса ячейки, к которой будет производиться доступ. В режиме считывания на входе 23 устанавливается Н-уровень, в режиме записи — L-уровень, а записываемые данные (m-разрядное слово) подаются на входы 24. На входе 28 пока присутствует Н-уровень.

Регистр 2, как уже отмечалось, при наличии Н-уровня на входе 28 транслирует код с входов 22 на выходы 33, 34, 36. Разряды А11, А10, А9 адреса содержат информацию о номере накопительного блока 1_1—1_8, который будет активизирован. С помощью этих разрядов мультиплексор 13 настраивается на передачу в точку 37 сигнала с выхода одного из компараторов 9_i, соответствующего блоку 1_i, к которому будет производиться обращение.

Компаратор 9_i сравнивает номер страницы, поступивший с линий 33, с номером, хранимым в регистре 10_i (из предыстории). В рассматриваемом режиме номера указанных страниц совпадают, так что в точке 37 устанавлнвается L-уровень. Это — первое необходнмое условие выполнения сокращенного цикла, которое проверяется блоком управления 8.

Вторым необходимым условием, прн котором возможно выполнение сокращенного цикла, является наличие (из предыстории) активного уровня (L) на входе 46_i (RAS) выбранного накопительного блока. Это условие также проверяется блоком управления 8, на вход 47 которого поступает информацня о состоянии выбранного мультиплексором 14 сигнала RAS. Мультиплексор 14 выбирает нужный сигнал RAS с выходов регистра 15, который отражает предысторию работы блоков 1_1—1_8: те блокн, к которым ранее производнлось обращение, «помечены» в регистре 15 сигналами L-уровия, а «невостребованные» блоки — сигналами Н-уровня.

Таким образом, на начальном этапе работы устройства в режнме сокращенного цикла на входах 37 и 47 блока управления 8 прнсутствуют L-уровни, а на входах 29, 41 и 42 — Н-уровни (пока предполагаетси, что сигнал запроса на регенерацню в линии 41 отсутствует).

После окончания переходных процессов в точках 37 н 47 на вход 28 устройства поступает сигнал L-уровня, подтверждающий истинность информации, ранее установленной на входах 22, 23, и, если выполняется операция записи, — на входах 24. Сигнал L-уровня «защелкивает» в регистре 2 информацию, которая ранее через него транслировалась, и, пройдя через элемент НЕ 12, переводит триггер 19 в единичное состояние, при котором на его нулевом выходе 42 формируется активный сигнал (L) запроса на обслуживание процессора.

По положнтельному фронту сигнала С1 с выхода генератора 62 (см. рис. 7.60, 7.66) запрос на обслуживанне процессора фиксируется в регнстре 63. Начиная с этого момента (t_0) вступает в дейст-

вие блок управления 8 (ранее он в каждом цикле работы «подтверждал» единичные состояния всех разрядов регистра 64).

При прохождении через логический блок (см. рис. 7.61) входные сигналы L-уровня с линий 69, 70 и 72 инвертируются элементами НЕ 96—98 и в виде H-уровней поступают на входы элемента И—НЕ 102. На один из входов этого элемента поступает H-уровень с выхода элемента НЕ 100, на вход которого подан L-уровень с выхода дешифратора 113, который в данном случае распознает комбинацию ННН на своих входах 93—95. На выходе элемента И—НЕ 102 формируется L-уровень, который проходит через элементы И 105 и 106 на выходы 75 и 80 логического блока 68.

Таким образом, сигналы L-уровней, поступившие на входы 69, 70, 72 логического блока, проходят через него и в виде L-уровней поступают на входы разрядов 85 и 90 регистра 64, в котором они запоминаются по отрицательному фронту сигнала С1 (см. рис. 7.66, момент t_1).

Сигнал L-уровня RCAS с выхода разряда 85 регистра 64 проходит через элемент И 65 в линию 26, в результате открывается дешифратор 16 и на линии 48_i, соответствующей выбранному накопительному блоку, формируется сигнал CAS (что и требуется в страничном режиме). Одновременно с этим формируется отрицательный фронт сигнала на выходе 27 ответа устройства и устанавливается в 0 триггер 19 — на его нулевом выходе 42 формируется H-уровень.

Комбинация сигналов LHH на выходах Т0—Т2 регистра 64, сформированная в момент t_1, не распознаётся дешифратором 113, поэтому сигнал L-уровня снимается с его выхода ННН, а на всех остальных выходах 115—119 поддерживаются сигналы H-уровней.

В момент t_2 сигнал H-уровня с входа 42 блока управления 8 переписывается в регистр 63.

В момент t_3 во всех разрядах регистра 64 фиксируются ранее сформированные логическим блоком H-уровни. Сигнал CAS снимается с выбранного накопительного блока; на выходе 27 ответа от устройства формируется положительный фронт сигнала, по которому (при считывании) можно «защелкивать» считанные из ОЗУ данные во внешнем регистре (не показан), подключенном к линиям 25. При записи в ОЗУ положительный фронт ответного сигнала на выходе 27 подтверждает факт выполнения этой операции.

В момент t_4 блок управления находится в исходном состоянии ожидания поступления сигнала запроса от процессора или от генератора 7, задающего частоту регенерации.

Отметим, что на протяжении всего рассмотренного цикла блок 5 транслировал младшие разряды адреса А8—А1 в шину 35, а блоки 3 и 4 оставались выключенными; информация в регистре 15 не менялась.

Режим полного цикла. Этот режим соответствует начальным участкам временных диаграмм, приведенных на рис. 7.63, 7.64. В режиме полного цикла в БИС 51_1—51_m выбранного накопительного блока передается адрес строки и адрес столбца по фронтам сигналов RAS и CAS, причем сигнал RAS в конце цикла не снимается.

Необходимость выполнения полного цикла возникает в двух ситуациях:

а) когда в предыстории отсутствовал активный уровень (L) на входе 46_i (RAS) выбранного (i-го) накопительного блока. Это, в свою очередь, означает, что обращение к выбраному накопитель-

ному блоку производится впервые после очередного цикла регенерации или начальной установки;

б) при наличии (из предыстории) активного уровня (L) на входе 46_i (RAS) выбранного (i-го) накопительного блока и несовпадении номера текущей страницы с номером предыдущей. Это означает, что блок 1_i ранее был выбран, но старшие 8 разрядов адреса, сопровождавшие предыдущее обращение, не совпадают с темн, которые поступили в регистр 2 при текущем обращении.

В первой ситуации на вход 47 блока управления 8 поступает Н-уровень (нет сигнала RAS на входе 46_i выбранного накопительного блока 1_i), причем сигнал Hit на входе 37 блока 8 может иметь произвольное значенне. При поступлении уровня L на вход 70 логнческого блока 68 на выходе элемента НЕ 97 формируется Н-уровень. В то же время сигнал Н-уровня с входа 72 блока 68 проходит через элемент НЕ 98 и в виде сигнала L-уровня воздействует на элемент И—НЕ 101, на выходе которого формируется Н-уровень. Так как на входы 93—95 блока 68 в исходном состоянии поданы Н-уровни, на линии НННН сформирован L-уровень, который проходит через элемент НЕ 100 и в виде Н-уровня поступает на вход элемеита И—НЕ 103, в результате он формирует на выходе сигнал L-уровня, который проходит на выходы 77 ($SRAS_i$), 74 (а) и 82 (Т2) блока 68.

Во второй ситуации (см. рис. 7.67) начальная реакция блока управления 8 — та же самая: L-уровни формируются на выходах 77, 74 и 82 блока 68. Действнтельно, в момент t_0 запуска блока управления 8 совпадение страниц не зарегистрировано (Hit=H), хотя сигнал RAS_{IN}=L. В блоке 68, как и в первой снтуацин, Н-уровнн поступают на все входы элемента И—НЕ 103, что приводит к формнрованию сигналов L-уровня на выходах $SRAS_i$, а и Т2 (см. рис. 7.67, момент t_1).

Сигнал L-уровня а с выхода 32 блока 8 проходит через элемент И—НЕ 21 и выключает блок передающих элементов 5, вместо которого включается блок 4, который транслирует в шину 35 старшие разряды адреса А19—А12. В то же время сигнал L-уровня с выхода 39 ($SRAS_i$) блока управления 8 открывает дешифратор 18, который формирует сигнал установки единицы выбранного разряда регистра 15. Сигнал RAS на входе 46_i выбранного блока 1_i переходнт в состояние H, если он находился в состоянии L (вторая из рассмотренных ранее снтуаций); если же он уже находился в состоянии Н (первая снтуацня), то его состояние не изменяется. Сигнал L-уровня с выхода дешифратора 18, кроме того, передается через элемент НЕ 11_i на управляющий вход регистра 10_i, который открывается по входам, готовясь в дальнейшем зафиксировать номер новой страницы. При этом компаратор 9_i регистрирует совпадение кодов, что, однако, уже не влияет на дальиейшую последовательность «микрокоманд», генерируемых блоком управления 8.

Информация, зарегистрированная в разрядах 90—92 регистра 64 в момент t_1 (HHL), однозначно определяет дальнейшее «поведение» блока 8. Действительно, указанная комбинация распознаётся дешифратором 113, в результате на его выходе 117 формируется L-уровень, который проходит на выходы 74 (а), 76 ($RRAS_i$) и 81 (Т1), см. рис. 7.67 момент t_2. В момент t_2 устанавливается в состояние L выбранный дешифратором 17 i-й разряд регистра 15, в результате на выбранный блок 1_i подается сигнал RAS (L-уровень), а регистр 10_i «защелкивает» старшие разряды адреса, так как на выходе элемента НЕ 11_i установился L-уровень.

Комбинация HLH в разрядах T0, T1, T2 регистра 64 распознаётся дешифратором 113, на выходе 118 которого формируется L-уровень, который проходит на выходы 81 (T1) и 82 (T2) блока 68. В момент t_3 эта кодовая комбинация фиксируется в регистре 64. Сигнал a принимает значение H, что приводит к включению группы передающих элементов 5 и к выключению группы 4 (проведена подготовка к передаче младших разрядов адреса в БИС 51_1—51_m выбранного накопительного блока 1_i).

Комбинация HLL вызывает формирование L-уровня в линии 115, который проходит на выходы 75 (RCAS) и 80 (T0) блока 68 и фиксируется в регистре 64 в момент t_4. При этом открывается дешифратор 16 и производится выдача сигнала CAS (L-уровнем) на выбранный накопительный блок. На выходе 27 формируется ответный сигнал, триггер 19 устанавливается в 0.

Комбинация LHH, сформированная в разрядах T0, T1, T2 регистра 64 в момент t_4, не распознаётся дешифратором 113, поэтому в момент t_5 на всех выходах блока управления 8 формируются H-уровни, т. е. устройство переходит в исходное состояние.

Режим регенерации. При поступлении положительного фронта сигнала с генератора импульсов 7 на вход синхронизации триггера 20 на его нулевом выходе 41 устанавливается напряжение низкого уровня — сигнал запроса на регенерацию. В момент t_0 (см. рис. 7.68) запрос принимается в регистр 63. При отсутствии запроса от процессора (MS=H) блок управления 8 вырабатывает последовательность сигналов, обеспечивающих выполнение цикла регенерации (см. рис. 7.65) одновременно во всех БИС 51_j всех накопительных блоков 1_i.

В момент t_0 на входы 70 и 71 блока 68 действуют сигналы H и L, причем на выходе HHH дешифратора 113 в исходном состоянии сформирован сигнал L, так как на входы 93—95 поданы H-уровни. Это приводит к формированию L-уровня на выходе элемента И—НЕ 104, который поступает на выходы 73 (b), 79 (SRG), 80 (T0), 81 (T1) и 82 (T2). Эти сигналы в момент времени t_1 фиксируются в регистре 64.

Сигнал SRG (L-уровень) подается на линию 40 и вызывает сброс триггера 20 и установку в единичное состояние (H) всех разрядов регистра 15 (все сигналы RAS, накопившиеся ранее в регистре 15, снимаются). Сигнал b передается на линию 31 и вызывает отключение от шины 35 группы передающих элементов 5 и подключение к этой шине группы передающих элементов 3, которые транслируют в шину сигналы с выхода двоичного счетчика 6, содержимое которого указывает номер строки, подлежащей регенерации.

Сигналы L-уровня, сформированные в момент t_1 в разрядах 90—92 регистра 64, однозначно определяют комбинацию сигналов, формируемых блоком 68 к моменту t_2: на выходе 116 дешифратора 113 формируется L-уровень, который поступает на выходы 73 (b), 78 (RRG), 80 (T0), 82 (T2).

Сигнал RRG (L-уровень) подается на линию 44 и вызывает формирование L-уровней на всех выходах регистра 15. При этом сигналы RAS (L-уровни) поступают на все БИС 51_j всех накопительных блоков 1_i, что приводит к регенерации очередной строки.

Сигналы L-уровней, сформированные в разрядах 90 и 92 регистра 64 в момент t_2, распознаются дешифратором 113. На его выходе 119 формируется L-уровень, который поступает на выходы 79 (SRG) и 80 (T0). Эти сигналы запоминаются в регистре 64 в момент t_3.

Сигнал SRG (L-уровень) поступает в линию 40 и устанавливает в состояние Н все разряды регистра 15. Сигнал b на линии 31 переходит в состояние Н, что вызывает прибавление единицы к содержимому двоичного счетчика 6, выключение группы элементов 3 и включение группы элементов 5. Сигналы LHH с выходов 90—92 регистра 64 поступают на дешифратор 113, но не распознаются им, в результате в момент t_4 блок управления 8 переходит в исходное состояние.

При одновременном поступлении в регистр 63 запроса от процессора и генератора 7 в первую очередь обслуживается запрос от процессора, а затем — запрос на регенерацию. Этот порядок может быть изменен на обратный при соответствующем изменении структуры логического блока 68. Необходимость такого изменения может возникнуть, когда запросы от процессора (процессоров) поступают слишком часто, «оттесняя» запрос регенерации. Более подробно об этой проблеме см. § 7.30.

В приведенном описании не рассмотрены режимы работы ОЗУ, не имеющие отношения к существу предлагаемого решения. К таким режимам относятся: Запись байта, Чтение — модификация — запись, Чтение с коррекцией одиночных и обнаружением двойных ошибок и др. Для реализации этих режимов достаточно увеличить число входов и выходов блока управления 8, число микропрограмм, выполняемых блоком 8, и внести в схемы необходимые изменения.

Благодаря расслоению адресов, поступающих в ОЗУ из процессора, фрагменты программ и данных, которыми оперируют микро-ЭВМ, оказываются перемешанными, а полученная смесь — равномерно «разлитой» в разные физические блоки памяти. Чем больше имеется таких блоков, тем меньше вероятность конфликтов, при которых процессор попеременно извлекает команды и данные из одного и того же блока, но с разных его страниц. (Напомним, что в отсутствие конфликтов используется страничный режим работы микросхем памяти.) Операционная система в данном случае не заботится о том, как распределить память между программой и данными для создания условий работы микросхем памяти в страничном режиме.

Если же возложить на операционную систему заботу о том, чтобы программы и массивы данных находились в физически разных блоках памяти, то можно практически исключить конфликты и, следовательно, в полной мере использовать преимущества страничного режима работы микросхем памяти [209].

7.25. Управление программным семафором без использования режима Чтение — модификация — запись

Постановка задачи и два варианта ее решения были рассмотрены в § 2.1. На рис. 7.70 приведен алгоритм решения задачи применительно к простейшим системам, у которых в интерфейсе не предусмотрен режим Чтение—модификация—запись.

Идея построения этого алгоритма состоит в том, что «выигрывает последний вошедший». Конкуренты записывают свои собственные номера $N_c \neq 0$ в ячейку S (семафор), если, по их сведениям, семафор открыт (S=0). Этот процесс неустойчив и не может длиться бесконечно, так как первый же конкурент закрывает семафор

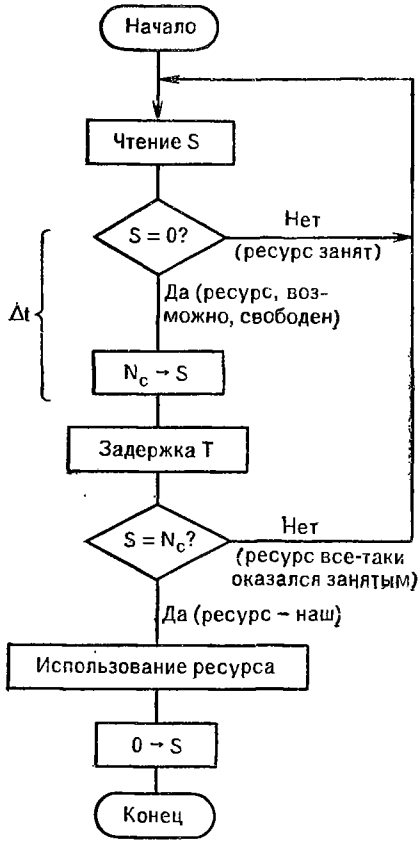


Рис. 7.70. Блок-схема алгоритма обращения устройства-конкурента к общему ресурсу с использованием программного семафора

записью собственного номера в ячейку S. Через некоторое время T участник конкурса (из числа тех, кому «посчастливилось» записать свой номер в ячейку S) проверяет содержимое этой ячейки и «узнаёт», является ли он победителем. Время T должно быть бо́льшим максимального времени установления окончательного кода в ячейке S. Максимальное время, в свою очередь, определяется длительностью выполнения цепочки операций, отмеченных на рис. 7.70 фигурной скобкой, всеми n конкурентами, которые успели «увидеть» разрешающий сигнал семафора перед моментом его закрытия ($T = n\Delta t$, где Δt — время выполнения цепочки команд, отмеченных на рис. 7.70).

Приведенный алгоритм, однако, связан с большими временными затратами на поиск победителя в конкурсе, что равносильно тому, что общий ресурс, например устройство для быстрого перемножения кодов, становится «медленным» (и может быть уже нет смысла вообще его использовать). Поэтому область применения данного алгоритма ограничена.

Отметим, что в этом и предыдущих алгоритмах (см. § 2.1) устройство-победитель может проявить «гуманность» и передать право пользования ресурсом другому устройству, просмотрев таблицу приоритетов, реализованную в своей или общей памяти.

7.26. Использование динамического ОЗУ для хранения констант, «не разрушающихся» при выключении напряжения питания

Микросхемы динамической памяти, например типа К565РУ5 (64К×1 бит), К565РУ7 (256К×1 бит), обладают примечательным свойством: при отключении регенерации информация в БИС теряется, однако 50 % запоминающих элементов постепенно переходят в состояние лог. 1, а остальные 50 % — в состояние лог. 0 (см. § 6.16, табл. 6.6). Такой же переход обнаруживается в ОЗУ или его части после включения напряжения питания, если задержать включение регенерации ОЗУ или его части примерно на 30 с.

Если прочитать содержимое ОЗУ (рис. 7.71, коммутаторы $К_i$ пока не принимаем во внимание) после включения напряжения пита-

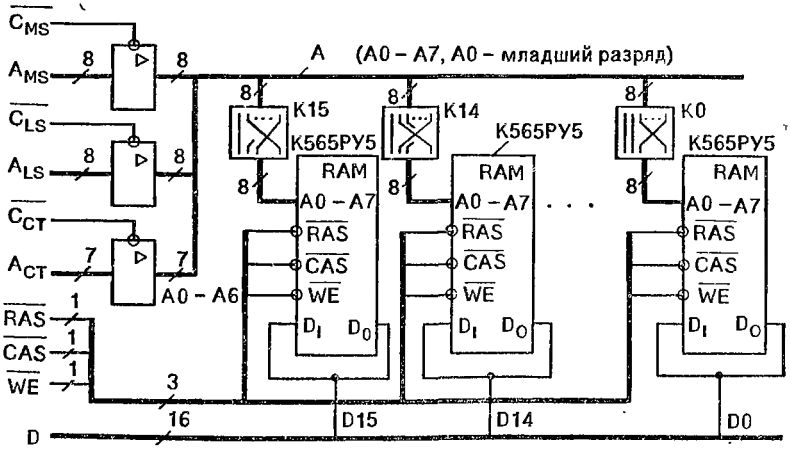


Рис. 7.71. Схема подключения к ОЗУ со структурой 64К×16 бит коммутаторов $К_i$, реализующих функции перестановки разрядов А0—А6 адреса.

Появляется возможность программировать содержимое одной или нескольких ячеек «скрытого ПЗУ»; содержимое этих ячеек не разрушается при отключении напряжения питания ОЗУ

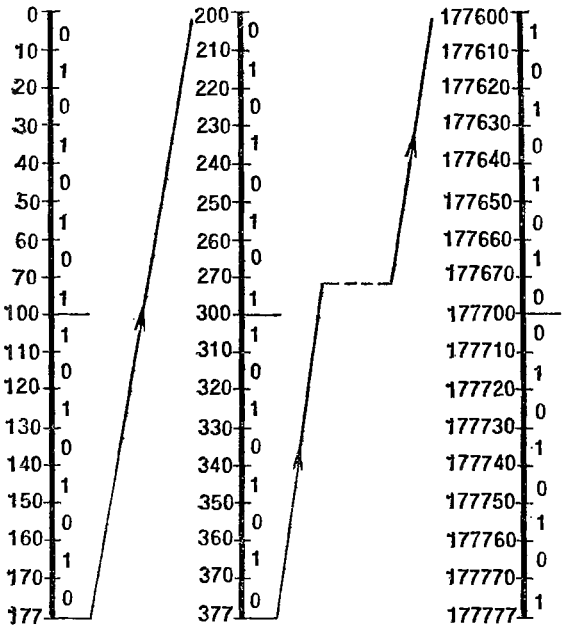


Рис. 7.72. Информация, формируемая в ОЗУ (см. рис. 7.71, без учета коммутаторов K_1) после отключения регенерации.

Закономерность распределения нулей и единиц определяется внутренней структурой БИС К565РУ5

ния и соответствующей задержки включения регенерации (до обращения к ОЗУ со стороны процессора), то в нем будет обнаружена чередующаяся последовательность групп нулевых и «единичных» кодов, как показано на рис. 7.72. В ячейках 0, 1, 2, ..., 7 присутствуют нулевые коды (при считывании — напряжения H высокого уровня), в ячейках 10, 11, 12, ..., 17_8 — коды 177777_8 (при считывании — напряжения L низкого уровня), условно обозначенные на рисунке символом 1, и т. д. (Рис. 7.72 соответствует столбцу 15 табл. 6.6 и его «продолжению» в сторону возрастания адресов; отличия состоят в адресации: в табл. 6.6 ячейкам соответствуют только четные адреса, а на рис. 7.72 использованы как четные, так и нечетные адреса, что не принципиально.)

Таким образом, в динамическом ОЗУ после включения напряжения питания и соответствующей задержки включения регенерации формируется определенная информация и в этом смысле можно считать, что динамическое ОЗУ содержит в себе некое «скрытое ПЗУ» с определенной кодировкой.

Представляет интерес вопрос о возможности кодирования заданной пользователем информации в одной или нескольких ячейках

«скрытого ПЗУ». Подтверждение этой возможности демонстрируется на приведенных далее примерах.

Чтобы понять существо предлагаемых решений, рассмотрим сначала работу схемы, приведенной на рис. 7.71, предполагая, что коммутаторы K_I транслируют сигналы с шины A на адресные входы БИС без «перекрестий» разрядов (т. е. фактически коммутаторы K_I исключены из схемы). При обращении к некоторой 16-разрядной ячейке ОЗУ в шину A адреса сначала поступают старшие A_{MS}, а затем младшие A_{LS} разряды полного 16-разрядного адреса. Передача старших и младших разрядов адреса сопровождается соответственно сигналами $\overline{RAS}$ и $\overline{CAS}$. При записи $\overline{WR}$=L (напряжение низкого уровня), информация с шины D поступает на входы D_I БИС и запоминается в них, при чтении $\overline{WR}$=H (напряжение высокого уровня), информация с выходов D_O БИС поступает в шину D.

Через каждые 15 мкс проводится регенерация очередной строки накопительных матриц, размещенных внутри БИС. Для этого очередной код A_{CT} с 7-разрядного двоичного счетчика (не показан) проходит в шину A и сопровождается сигналом $\overline{RAS}$, в конце цикла к коду A_{CT} прибавляется единица и т. д. Сигналы $\overline{C}_{MS}$, $\overline{C}_{LS}$ и $\overline{C}_{CT}$ служат для включения соответствующих групп усилителей в нужные промежутки времени; в отсутствие этих сигналов группы усилителей выключены.

С помощью коммутаторов K_I (рис. 7.73) можно «переставить» разряды адреса A0—A6, не меняя положение разряда A7. Разряд A7 не используется при регенерации, в то время как в разрядах A0—A6 с помощью двоичного счетчика последовательно создаются все возможные кодовые комбинации, которые, проходя через коммутатор, обеспечивают регенерацию всех строк накопительных матриц. С учетом прохождения сигналов через коммутатор порядок следования номеров строк, подлежащих регенерации, может отличаться от «естественного», что, однако, не имеет значения.

Перестановка разрядов A0—A6 адреса с помощью коммутатора не нарушает работоспособности ОЗУ. Действительно, благодаря однозначности преобразования адреса не возникает каких-либо проблем при записи и считывании информации — считываются именно те биты, которые были ранее записаны (см. § 7.1). В то же время перестановка разрядов адреса приводит к изменению кодировки «скрытого ПЗУ».

Предположим, что необходимо записать в ячейку 22_8 «скрытого ПЗУ» код 123456_8. Из рис. 7.72 следует, что после включения напряжения питания в ячейке 22 устанавливается нулевой код. Чтобы получить единицы в нужных разрядах, достаточно для соответствующих БИС преобразовать адрес 22, например, в адрес 30, которому соответствует «единичная» информация (см. рис. 7.72). Оба адреса содержат по две единицы: 22_8=10010, 30_8=11000, поэтому второй может быть получен из первого перестановкой разрядов, как показано на рис. 7.74.

Из рис. 7.74 следует, что существует широкий выбор структур коммутаторов, обеспечивающих считывание «единицы» из ячейки с адресом 22, преобразованным в адрес 30. (Отметим, что адрес 22 можно было бы преобразовать в адрес 11, 12, 14, 50 и др.) Выбираем один из коммутаторов (любой). Схема подключения БИС ОЗУ

к шине А с использованием коммутаторов выбранного типа показана на рис. 7.75. Коммутаторы К включены в тех разрядах слова, из которых должна быть считана лог. 1. Из тех разрядов, где коммутаторов нет, после включения напряжения питания считываются лог. 0. Таким образом, с «точки зрения» процессора, после включения напряжения питания в ячейке 22 будет сформирован код 123456_8, что и требовалось.

Рис. 7.73. Схема коммутатора.

Коммутатор производит перестановку разрядов А0 — А6 при передаче адреса с шины А на входы БИС; разряд А7 не участвует в перестановках

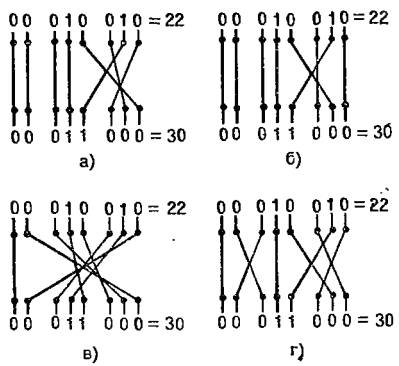


Рис. 7.74. Варианты построения коммутаторов, преобразующих код 22 в код 30

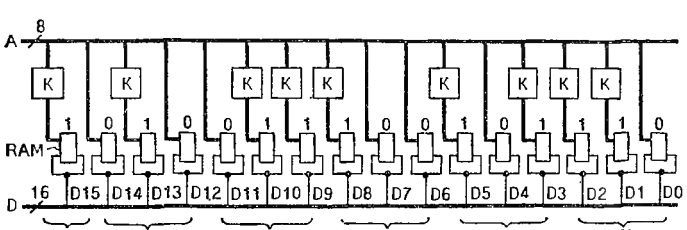
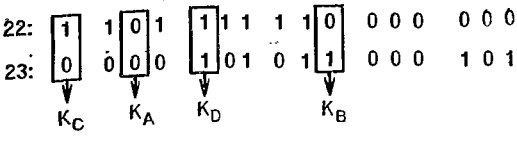


Рис. 7.75. Схема подключения к ОЗУ коммутаторов К, преобразующих код 22 в код 30 (рис. 7.74).

При таком подключении в ячейке 22 «скрытого ПЗУ» хранится код 123456_8 (из ячеек с адресом 22 считываются 0, а из ячеек с адресом 30 — единицы)

Покажем теперь, как синтезировать информацию в двух соседних ячейках «скрытого ПЗУ». Предположим, что в ячейках 22 и 23 необходимо получить коды 157600_8 и 005305_8:



Коды в ячейках 22 и 23 различны. В каждом разряде при переходе от ячейки 22 к ячейке 23 наблюдается один из переходов: 0→0, 0→1, 1→0, 1→1, см. разряды, выделенные «рамками». Можно предположить, что существуют 4 типа коммутаторов K_A, K_B, K_C и K_D, обеспечивающих эти переходы. Определим структуру этих коммутаторов.

При включении напряжения питания в ячейках 22 и 23 «обычного» ОЗУ устанавливаются нулевые коды (см. рис. 7.72). Следовательно, переход 0→0 уже «заложен» в основной структуре ОЗУ. Иными словами, в тех разрядах, где должен быть такой переход, коммутатор K_A вообще не нужен (можно считать, что он есть, но транслирует сигналы без «перекрестий»).

Примеры построения коммутаторов K_B, K_C и K_D приведены на рис. 7.76. Структуры коммутаторов получены путем ручного перебора вариантов до первого удачного; выбор приемлемых решений не составляет труда. Коммутатор K_B транслирует адрес 22 без изменения, что обеспечивает считывание 0 по этому адресу (см. рис. 7.72). Однако адрес 23 преобразуется коммутатором K_B в адрес 32, по которому, согласно рис. 7.72, хранится 1. Коммутатор K_C преобразует адрес 22 в адрес 11, что обеспечивает считывание 1, а адрес 23 — в адрес 111, по которому записан 0. Коммутатор K_D преобразует адрес 22 в адрес 30 и адрес 23 в адрес 31; согласно рис. 7.72 оба преобразованных адреса обеспечивают считывание единиц.

Схема, обеспечивающая хранение констант 157600 и 005305 в ячейках 22 и 23 (в отсутствие напряжения питания ОЗУ), приведена на рис. 7.77. В ней использованы коммутаторы трех типов: K_B, K_C и K_D

Аналогично можно синтезировать информацию в большем числе ячеек «скрытого ПЗУ», однако задача выбора структур коммутаторов при этом существенно усложняется. «Абсолютный рекорд» здесь не известен, но, вероятнее всего, он не превышает десятка ячеек. Отметим, что не все ячейки могут быть использованы для синтеза кодов; самыми неудачными являются ячейки с адресами 000...0 и 111...1. Перестановка адресных разрядов каждой из этих ячеек не дает эффекта — ячейки «преобразуются сами в себя». Среди остальных ячеек также есть «худшие» и «лучшие» с точки зрения возможностей синтеза заданных групп кодов.

Возможность хранения заданных констант в ОЗУ в отсутствие напряжения питания может оказаться полезной во многих ситуациях. Например, константа может содержать информацию о характеристиках системы, в которой используется микроЭВМ, задавать приоритет и собственный номер микроЭВМ в многомашинной системе, определять возможность коллективного пользования данной машиной и т. п. При использовании микропереключателей в составе коммутаторов программа может по полученной при считывании кодировке «скры-

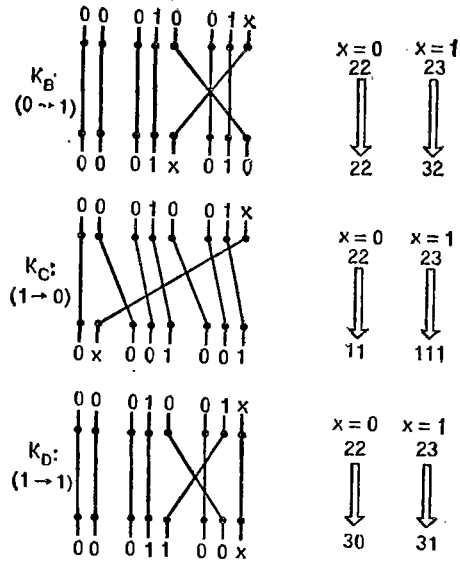


Рис. 7.76. Варианты построения коммутаторов K_B, K_C и K_D

того ПЗУ» определить состояние этих микропереключателей (аппаратные затраты на опрос состояния микропереключателей равны нулю).

Чтобы не вводить в ОЗУ схему задержки включения регенерации при подаче напряжения питания, можно производить регенерацию не всех строк накопительных матриц. При этом некоторая область ОЗУ не будет регенерироваться и в ней примерно через 30 с после включения напряжения питания проявится кодировка «скрытого ПЗУ», которая в дальнейшем доступна по чтению в произволь-

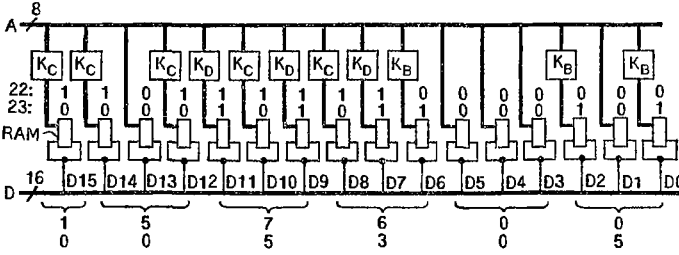


Рис. 7.77. Подключение к ОЗУ коммутаторов (K_B, K_C и K_D), обеспечивающее задание кодов 157600 и 005305 в ячейках 22 и 23 «скрытого ПЗУ»

ные моменты времени с темпом, свойственным ОЗУ. Не регенерируемая область ОЗУ может использоваться только для хранения констант, размещенных в соответствующей области «скрытого ПЗУ». При попытке записи в эту область новой информации последняя будет постепенно исчезать и «превращаться» в кодировку «скрытого ПЗУ», если нет периодического обращения процессора к соответствующим строкам этой области (т. е. если нет «программной регенерации»).

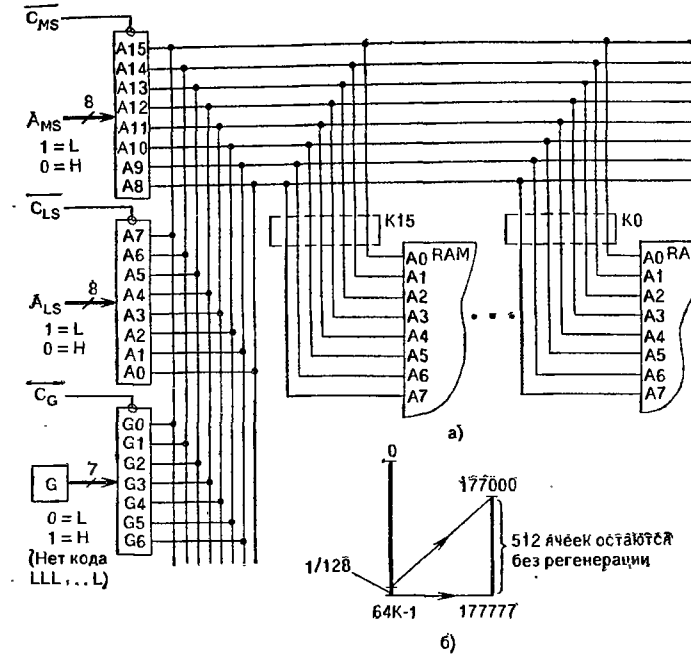


Рис. 7.78. Схема ОЗУ.

При использовании генератора G псевдослучайной последовательности кодов (*а*) одна из строк в каждой БИС ОЗУ не регенерируется, в результате последние 512 ячеек ОЗУ (*б*) «проявляют» информацию, записанную в «скрытом ПЗУ»

В примере схемного решения (рис. 7.78, *а*) для формирования адресов строк, подлежащих регенерации, использован генератор G псевдослучайной последовательности кодов. Генератор вырабатывает циклическую последовательность 7-разрядных кодов, в которой присутствуют все коды, за исключением нулевого. Таким образом, одна из 128 строк накопительных матриц БИС не регенерируется и 512 ячеек ОЗУ «проявляют» информацию, записанную в соответствующих 512 ячейках «скрытого ПЗУ». Распределение разрядов в шине А выбрано так, что область ОЗУ с размещенными в ней кон-

стантами «скрытого ПЗУ» представляет собой непрерывный массив ячеек, расположенный в зоне старшнх адресов, как показано на рис. 7.78, *б*. Для получения нужных констант в схему, приведенную на рис. 7.78, *а*, вводятся соответствующие коммутаторы.

Применение микросхем К565РУ7 позволяет расширить возможности выбора структур коммутаторов, так как регенерация накопительных матриц этих микросхем производится без использования «внешнего» счетчика. Поэтому в «перестановках» могут участвовать все 9 разрядов адресной шины.

7.27. МикроЭВМ с программируемыми адресами контроллеров [135, 179]

Адреса контроллеров внешних устройств обычно задаются с помощью микропереключателей или проводных перемычек, которые устанавливаются на плате. Как показано в примере, приведенном на рис. 7.79, код A*, задаваемый микропереключателями или перемычками, с помощью компаратора EQ1 сравнивается с группой разрядов адресного кода, поступающего из центрального процессора ЦП по адресной шине A магистрали. При совпадении кодов сигнал с выхода компаратора EQ1 активизирует контроллер K1. В зависимости от сигналов на линиях I/O R (чтение информации из устройства ввода-вывода) и I/O W (запись информации в устройство ввода-вывода) контроллер K1 считывает или записывает информацию в один из внутренних регистров, выбираемых группой младших разрядов адреса, поступающих с шины A непосредственно в контроллер.

Недостаток такого способа задания собственных адресов контроллеров связан с неэффективным использованием площади печатной платы, так как переключатели и перемычки занимают много места. Кроме того, при «ручном» задании адресов возможны ошибки: неправильная установка перемычек или неправильный выбор положений микропереключателей.

Предлагается использовать программно-управляемое назначение собственных адресов контроллеров.

В микроЭВМ, показанной на рис. 7.79, использованы два изолированных друг от друга адресных пространства. Первое имеет объем, равный 1М байт; в нем размещены адреса ПЗУ и ОЗУ. Второе имеет объем, равный 1К байт, и предназначено для размещения в нем адресов регистров контроллеров. При работе с первым адресным пространством процессор формирует сигнал MEM R или MEM W — чтение или запись информации в ячейку памяти. Второму адресному пространству соответствуют сигналы I/O R и I/O W.

В персональных микроЭВМ типа IBM PC для ПЗУ выделены 64К ячеек первого адресного пространства: F0000—FFFFF. В зависимости от модели микроЭВМ ОЗУ занимает в первом адресном пространстве область от 00000 до 3FFFF или бо́льшую, до 9FFFF. В предлагаемом решении для хранения собственного адреса контроллера K2 используется программно-доступный регистр RG. Этот регистр доступен только по записи. Адрес самого регистра RG относится к первому адресному пространству и выбран равным FFF8E, т. е. совпадает с адресом одной из ячеек ПЗУ. Такое решение оказалось возможным благодаря тому, что ПЗУ доступно только по чтению, а регистр RG — только по записи. Так как сигналы MEM R и MEM W никогда не формируются одновременно, конфликтов меж-

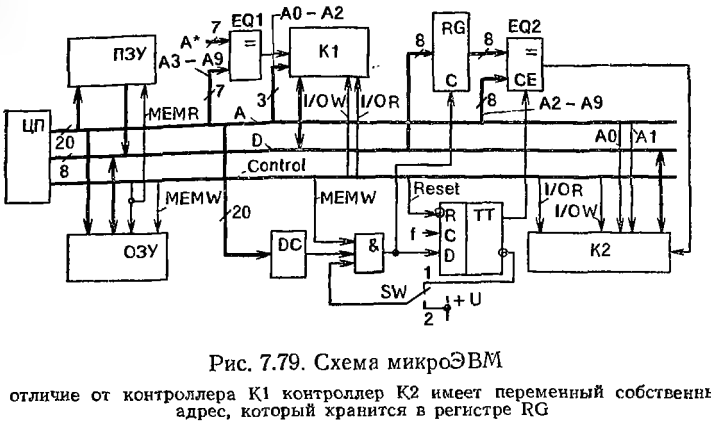


Рис. 7.79. Схема микроЭВМ

В отличие от контроллера К1 контроллер К2 имеет переменный собственный адрес, который хранится в регистре RG

ду ПЗУ и регистром RG быть не может, несмотря на то, что один и тот же адрес относится к обоим устройствам.

При включении напряжения питания микроЭВМ процессор выполняет программу начального пуска, в которой, в частности, предусмотрена загрузка регистра RG некоторым 8-разрядным кодом; в дальнейшем он сравнивается компаратором EQ2 с кодом в разрядах A2—A9 адресной шины A. При срабатывании компаратора активизируется контроллер К2. Для адресации устройств ввода-вывода в микроЭВМ используют 10 младших разрядов адреса A0—A9; разряды A0 и A1 поступают непосредственно в контроллер К2 и определяют выбор одного из четырех его внутренних регистров.

При загрузке регистра RG процессор выполняет команду пересылки содержимого одного из регистров общего назначения в ячейку памяти с адресом FFF8E. Этот адрес распознаётся дешифратором DC, в результате срабатывает элемент И и в регистр RG принимается 8-разрядный код с шины D. Триггер TT устанавливается в единичное состояние, элемент И закрывается, препятствуя повторной смене собственного адреса контроллера К2, например, из-за ошибки или преднамеренного действия программиста. Если ключ SW перевести в положение 2, то повторная смена адреса окажется возможной.

7.28. Ускорение записи неполноразрядных слов в память с большой разрядностью шины данных [136]

На рис. 7.80 показана функциональная схема блока памяти, обеспечивающего доступ к 64-разрядным ячейкам ОЗУ. Блок памяти содержит динамическое ОЗУ, в котором хранится $2^{20}=1M$ 64-разрядных слов (8M байт), 64-разрядные регистры RG1, RG2 для хранения байт 0—7, байтовые мультиплексоры M, блок передающих элементов TR и блок управления.

Регистры RG1 и RG2 принимают информацию по сигналам C1

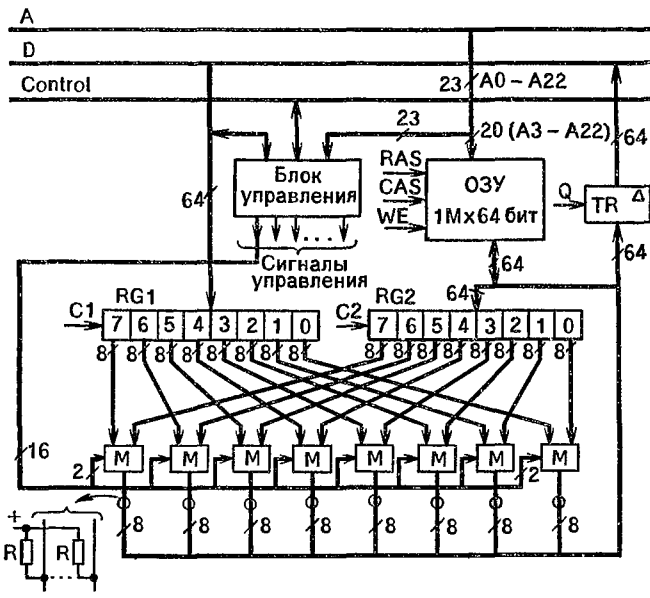


Рис. 7.80. Схема ОЗУ.

При записи неполноразрядного слова в 64-разрядную ячейку ОЗУ мультиплексоры М компонуют слово, содержащее «новую» и «старую» части, освобождая процессор от выполнения операций компоновки программным (микропрограммным) способом

и С2, поступающим из блока управления. Каждый мультиплексор М имеет два управляющих входа, на которые поступают индивидуальные сигналы из блока управления. Один из управляющих входов предназначен для задания направления передачи информации через мультиплексор, второй вход позволяет перевести выходы мультиплексора в «третье состояние» (с высоким выходным сопротивлением), причем благодаря подключению резисторов R к выходам мультиплексоров на отключенных выходах поддерживаются сигналы лог. 1.

Блок передающих элементов TR в зависимости от сигнала Q на управляющем входе либо выключен, либо транслирует входные сигналы в 64-разрядную шину данных D магистрали. Сигналы RAS (выбор строки), CAS (выбор столбца) и WE (запись) поступают в ОЗУ из блока управления и задают режим его работы (чтение, запись, чтение — модификация — запись, регенерация).

При чтении 64-разрядного слова внешним абонентом, например процессором, 23-разрядный адрес поступает в адресную шину А магистрали, а код операции чтения — в линии шины управления Control. На адресные входы ОЗУ поступает 20-разрядный адрес, определяющий одну из 1М 64-разрядных ячеек (младшие разряды А0—А2 при выборе единой группы из восьми байт не нужны). Блок

управления формирует сигналы RAS, CAS и сигналом Q открывает передающие элементы TR; мультиплексоры M выключены. В результате считанное из ячейки ОЗУ 64-разрядное слово выдается в шину данных D магистрали и поступает в процессор. Если процессору нужны ие все байты считанного слова, то «лишние» байты просто теряются.

При записи 64-разрядного слова, поступающего из процессора в шину D, оно транслируется через регистр RG1, мультиплексоры M и поступает в выбранную 64-разрядную ячейку ОЗУ. Блок передающих элементов TR постоянно закрыт. Операции Чтение — модификация — запись 64-разрядного слова выполняется обычным путем: данные сначала считываются процессором, а затем после преобразования записываются в ту же ячейку ОЗУ, из которой производилось чтение.

Преимущества предлагаемого решения перед известными проявляется при выполнении операции записи, когда процессор меняет не весь 64-разрядный код в ячейке ОЗУ, а только его часть — от одного до семи байтов. Номер начального изменяемого байта (от нулевого до шестого) и число следующих за ним изменяемых байтов задаются соответствующими кодами, передаваемыми из процессора по шинам A (разряды A0—A2) и Control. Сами изменяемые байты передаются по соответствующим группам линий шины данных D.

Блок управления, распознав поступившую от процессора команду записи с неполной сменой информации в ячейке, вырабатывает определенную последовательность управляющих сигналов для выполнения следующих действий.

Из ячейки ОЗУ с заданным адресом считывается 64-разрядный код и запоминается в регистре RG2. (Передающие элементы TR на протяжеиии выполнения всей операции выключены.) Одновременно с этим код с шины данных D, содержащий одии или несколько (до семи) изменяемых байтов и произвольную информацию в остальных разрядах, запоминается в регистре RG1. Мультиплексоры M в соответствии с информацией, полученной по шинам A и Control, настраиваются блоком управления так, что изменяемые байты проходят через них из регистра RG1, а не изменяемые — из регистра RG2. В результате в ОЗУ поступает 64-разрядный код, в котором сохранена неизменяемая часть слова, а измеияемая содержит новую информацию, поступившую от процессора. Этот код запоминается в ячейке ОЗУ.

Регистр RG1 позволяет освобождать магистраль, не дожидаясь момента записи в ОЗУ результирующего кода. Если регистр RG1 исключить из схемы, то необходимо выполнить условие стабильности информации в шине D на протяжении всего цикла работы устройства, что замедляет работу процессора.

Устройство позволяет также производить быструю установку разрядов выбранного байта в 64-разрядной ячейке памяти в состояние 111...1. Эта операция бывает полезной при использовании выбранного байта в качестве программного «флага» для синхронизации нескольких вычислительных процессов. При выполнении операции процессор передает по шине A адрес с точностью до байта — разряды A0—A2 задают положение байта («флага») в 64-разрядном слове. Блок управления распознаёт тип операции по сигналам на шине Control и приступает к ее выполнеиию. Содержимое выбранной 64-разрядной ячейки ОЗУ запоминается в регистре RG2. Затем все мультиплексоры M, кроме одного, переводятся из выключенного

состояния в режим передачи кода из регистра RG2. На выходах мультиплексора, который остался выключенным (а именно он и соответствует «флаговому» байту) сформированы сигналы 111...1 за счет подключения резисторов R. Результирующий код записывается в выбранную 64-разрядную ячейку ОЗУ.

Отметим, что при подключении резисторов к шине нулевого потенциала можно получить код 000...0, а при «смешанном» подключении — любой заранее заданный «флаговый» код.

Таким образом, устройство позволяет освободить процессор от выполнения ряда рутинных операций при работе со словами переменной длины, а также при установке «флаговых» байтов в памяти.

7.29. Схема формирования ответных сигналов от абонентов, учитывающая различие их быстродействия [137]

При обращении процессора или другого активного устройства к адресуемому абоненту должен быть выработан ответный сигнал, подтверждающий, что требуемая операция (чтение, запись, чтение — модификация — запись, см. гл. 2) выполнена. Так как разные абоненты могут иметь разное быстродействие, ответные сигналы от абонентов должны формироваться с соответствующими (разными) задержками. Поэтому в каждом адресуемом устройстве предусматривают отдельную схему формирования ответного сигнала; при этом число таких схем в микроЭВМ оказывается равным числу абонентов.

Предлагается использовать единую схему для формирования всех ответных сигналов с учетом разного быстродействия абонентов (рис. 7.81).

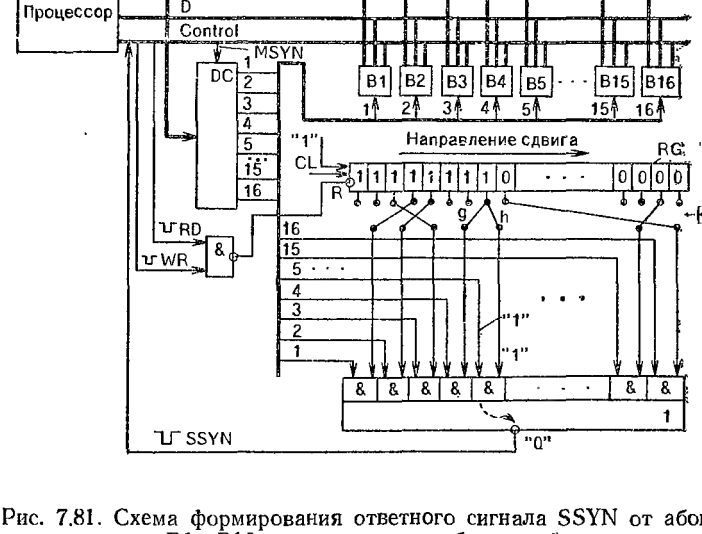


Рис. 7.81. Схема формирования ответного сигнала SSYN от абонентов B1—B16, имеющих разное быстродействие

При обращении процессора к одному из абонентов B_i на шине A магистрали устанавливается адрес абонента. Истинность адреса подтверждается сигналом MSYN на шине управления Control. С помощью дешифратора DC, анализирующего старшие разряды адреса, активизируется один из абонентов B_i, на который поступают младшие разряды адреса. По младшим разрядам в активизированном блоке B_i выбирается нужный адресуемый элемент (например, ячейка памяти). В зависимости от требуемой операции (чтение RD, запись WR или их последовательность RD—WR) данные либо выдаются из адресуемого элемента в шину D, либо принимаются с этой шины в адресуемый элемент, либо последовательно выполняются оба действия с одним и тем же элементом.

Ответный сигнал SSYN формируется элементом И—ИЛИ—НЕ следующим образом. Одновременно с выбором абонента сигнал лог. 1 с выхода дешифратора DC поступает на соответствующий вход элемента И—ИЛИ—НЕ, разрешая прием одного из сигналов с выходов сдвигового регистра RG. В исходном состоянии на всех выходах регистра RG сформированы сигналы лог. 0, так как на вход R установки нуля регистра подан сигнал лог. 0. При поступлении сигнала RD или WR на вход элемента И—НЕ регистр RG переходит в режим сдвига сигнала лог. 1, постоянно присутствующего на его входе.

Сдвиг производится с частотой следования импульсов CL, которая может быть равна, например, удвоенной тактовой частоте процессора. Сдвиговый регистр постепенно наполняется единицами (на рис. 7.81 показано одно из промежуточных состояний регистра). Когда появляется «нужная» единица, срабатывает элемент И—ИЛИ—НЕ и в процессор посылается ответный сигнал SSYN. При выполнении операции чтения или записи процессор освобождает магистраль, регистр переходит в исходное состояние.

При выполнении операции Чтение—модификация—запись адрес и сигнал MSYN не снимаются после снятия сигнала RD; при поступлении сигнала WR отсчет времени регистром RG начинается сначала и после выполнения записи в процессор посылается второй ответный сигнал SSYN. После этого процессор освобождает магистраль и регистр RG переходит в исходное состояние.

Выбор задержки формирования ответного сигнала SSYN производится подбором нужной перемычки на коммутаторе K. Если несколько абонентов имеют одинаковую задержку, то соответствующие входы элемента И—ИЛИ—НЕ соединяются с одним и тем же выходом регистра RG (см. перемычки g и h).

Если операции чтения и записи выполняются абонентом с разной задержкой, то этот факт можно учесть выделением двух (а не одной) точек подключения к регистру для этого абонента. При этом схема несколько усложняется, так как при формировании ответного сигнала следует дополнительно анализировать тип выполняемой операции (сигналы RD и WR).

7.30. Уменьшение потерь производительности микроЭВМ, связанных с регенерацией динамической памяти [138]

При работе динамического ОЗУ в составе микроЭВМ (рис. 7.82) необходимо примерно через каждые 15 мкс выполнять циклы регенерации, чтобы информация в ОЗУ не испортилась. Длительность

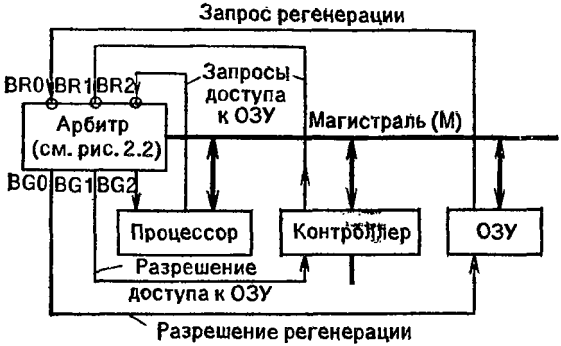


Рис. 7.82. Схема арбитража.

Запрос регенерации ОЗУ должен быть безусловно воспринят, поэтому ему обычно присваивают максимальный приоритет (вход BR0 арбитра); это, однако, приводит к неоправданным задержкам обслуживания запросов контроллера и процессора, имеющих более низкие приоритеты (входы BR1 и BR2)

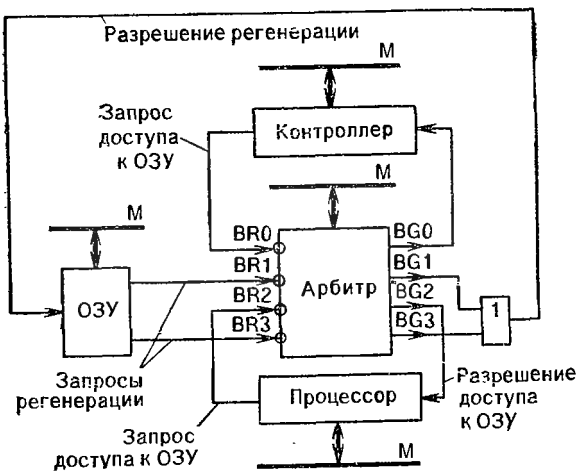


Рис. 7.83. Предлагаемая приоритетная структура микроЭВМ.

В отличие от контроллера и процессора, которым присвоены фиксированные приоритетные уровни, для ОЗУ выделены два уровня, которые используются по мере необходимости; М — магистраль

цикла регенерации составляет примерно 0,4 мкс. В течение цикла регенерации ОЗУ не может использоваться для выполнения «полезных» операций (чтение, запись, чтение — модификация — запись, запись бита, байта и др.), проводимых по инициативе процессора или контроллера прямого доступа в память.

Так как регенерация должна проводиться безусловно (иначе возникнут потери информации), запросу на регенерацию обычно присваивают максимальный приоритет при устранении конфликтов. Конфликты возникают из-за того, что процессор или (и) контроллер прямого доступа к памяти могут пожелать ею воспользоваться, в то время как назрела необходимость выполнения очередного цикла регенерации. Благодаря максимальному приоритету запрос регенерации удовлетворяется арбитром (см. гл. 2) в первую очередь. Обслуживание других запросов доступа к ОЗУ при этом откладывается примерно на 0,4 мкс. Итак, имеется противоречие: с одной стороны, запрос регенерации должен быть безусловно удовлетворен, с другой — из-за высокого приоритета запрос регенерации мешает обработке других запросов, откладывая их выполнение на 0,4 мкс.

Усовершенствование алгоритма работы управляющих схем ОЗУ и изменение приоритетной структуры микроЭВМ (рис. 7.83) позволяют обеспечивать гарантированное обслуживание запроса на регенерацию при уменьшении средней задержки обслуживания других запросов. Это оказалось возможным в силу следующих обстоятельств.

Цикл регенерации ОЗУ длительностью 0,4 мкс должен быть обязательно выполнен в течение интервала времени, равного 15 мкс. При этом не имеет значения, будет ли выполнен цикл регенерации в начале, середине или конце этого интервала. На протяжении следующих 15 мкс (в любое время) обязательно должен быть выполнен следующий цикл регенерации и т. д. Иными словами, не обязательно удовлетворять запрос регенерации сразу же после его поступления — лучше сначала обслужить «полезные» запросы, а когда магистраль окажется незанятой — обслужить запрос на регенерацию. Для этого приоритет запроса регенерации сначала устанавливается низким. Если же отведенный интервал времени (15 мкс) близится к концу, а магистраль попеременно (без пауз) используется процессором и контроллером для доступа к ОЗУ, то в остающееся время необходимо обязательно выполнить регенерацию. Для этого приоритет запроса регенерации должен быть повышен до уровня, обеспечивающего гарантированное обслуживание в течение оставшегося времени.

В схеме, показанной на рис. 7.83, максимальный приоритет присвоен запросу доступа к ОЗУ со стороны контроллера. Запросу от процессора присвоен более низкий приоритет, так что процессор всегда проигрывает в конкурентной борьбе с контроллером. Для запросов регенерации выделены два входа арбитра. В первой половине (7,5 мкс) интервала длительностью 15 мкс, отведенного для размещения в нем одного цикла регенерации, схемы управления ОЗУ формируют низкоприоритетный сигнал запроса на регенерацию, который поступает на вход BR3 арбитра. Запрос на входе BR1 арбитра отсутствует.

Если контроллер прямого доступа в память находится в пассивном состоянии или проявляет невысокую активность, то основным конкурентом запроса регенерации является запрос доступа к ОЗУ со стороны процессора, поступающий на вход BR2 арбитра (приоритеты входов возрастают с уменьшением номеров i входов BR_i).

Если бы процессор постоянно (без пауз) занимал магистраль, то запрос по входу BR3 арбитра не был обслужен. Однако процессор периодически освобождает магистраль, например, при выполнении арифметических операций над даиными, ранее считанными из ОЗУ. В течение первого «свободного» промежутка времени с помощью арбитра разрешается выполнение цикла регенерации. Таким образом, с появлением первой паузы в использовании магистрали проводится цикл регенерации и запрос от ОЗУ снимается до начала следующего интервала 15 мкс. Использование паузы позволяет проводить регенерацию «скрыто», без задержки работы процессора.

Если контроллер прямого доступа проявляет повышенную активность, то пауз в работе магистрали может не оказаться, следовательно, в течение первых 7,5 мкс запрос регенерации, поступающий на вход BR3 арбитра, не будет обслужен. В этом случае схемы управления ОЗУ формируют дополнительный запрос (первоначальный может и не сниматься), поступающий на вход BR1 арбитра. Таким образом, приоритет запроса регенерации становится более высоким, чем приоритет запроса процессора, но остается менее высоким, чем приоритет запроса контроллера прямого доступа в память. Контроллер, как предполагается, освобождает магистраль после каждого обращения к ОЗУ, так что в течение первой же паузы в работе контроллера осуществляется цикл регенерации и сигналы запросов от ОЗУ снимаются с входов арбитра.

Деление интервала 15 мкс на две равные части удобно при построении схемы управления ОЗУ. Одиако этот интервал можно делить, например, на три части по 5 мкс и нспользовать первые 5 или 10 мкс для формирования низкоприоритетного запроса, а оставшееся время — для формирования высокоприоритетного запроса регенерации. При необходимости можно и далее усовершенствовать схему (рис. 7.83), например, выделив для запросов регенерации три приоритетных уровня с переходом от одного к другому по мере истечения интервала времени, отведенного для выполнения цикла регенерации. Можно учесть и возможность еше более неравномерного распределения циклов регенерации во времени. В приведеином решении в качестве исходной предпосылки принималась безусловная необходимость выполнения одного цикла регенерации в течение каждых 15 мкс. Однако задачу можно сформулировать менее жестко, например, так: необходимо выполнить два цнкла регенерации за 30 мкс или три цикла за 45 мкс и т. д., размещая эти циклы произвольным образом.

7.31. Управление программными семафорами в многопроцессорной системе [139, 140, 186]

Чтобы понять постановку задачи, рассмотрим двухпроцессорную систему, показанную на рис. 7.84.

Процессор ЦП1 готовит в ОЗУ информацию для процессора ЦП2, который использует ее для дальнейшей обработки. Для врёменного хранения передаваемой информации в ОЗУ иыделена некоторая область — буфер, а для отображения числа имеющихся в буфере блоков информации (в простейшем случае, числа отдельных слов), выделена ячейка — счетчик. Буфер может иметь структуру типа LIFO («стек»), FIFO («трубопровод») или иную — в любом случае он является частью обычного ОЗУ. Код в счетчнке увеличи-

вается процессором ЦП1 на единицу после записи очередного блока информации в буфер. После считывания очередного блока информации из буфера процессором ЦП2 он уменьшает на единицу содержимое счетчика.

Предположим, что в этой системе наблюдается следующая последовательность событий:

а) ЦП1 считывает содержимое ячейки-счетчика;
б) ЦП1 прибавляет единицу к содержимому ячейки-счетчика;
в) ЦП1 записывает полученную сумму в ячейку-счетчик;
г) ЦП2 считывает содержимое ячейки-счетчика;

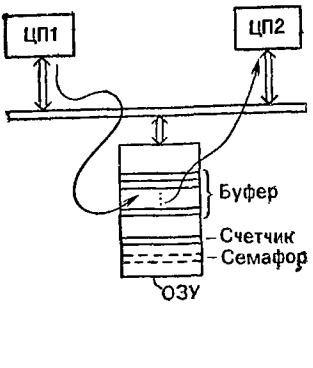


Рис. 7.84. Двухпроцессорная система с общей памятью

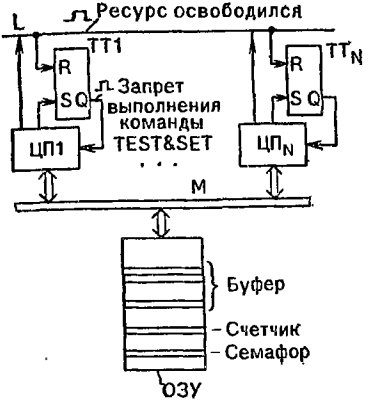


Рис. 7.85. Усовершенствованная многопроцессорная система с общей памятью.

Чтобы не создавать лишних пересылок информации по магистрали М, каждому процессору дается только одва попытка «взглянуть» на семафор; следующая попытка откладывается до момента освобождения ресурса тем процессором, который этим ресурсом пользовался

д) ЦП2 вычитает единицу из содержимого ячейки-счетчика;
е) ЦП2 записывает полученную разность в ячейку-счетчик.

Данная последовательность событий соответствует бесконфликтному взаимодействию процессоров ЦП1 и ЦП2: ЦП1 пополняет буфер очередным блоком информации, а ЦП2 извлекает из буфера некоторый блок. При этом содержимое ячейки — счетчика сначала увеличивается, а затем уменьшается на единицу, т. е. в результате остается без изменения, что и ожидалось.

Иные (ошибочные) ситуации возникают при более «тесном» взаимодействии процессоров ЦП1 и ЦП2. Приведем в качестве примера такого взаимодействия две возможные последовательности ра-

нее описанных событий и соответствующие им числа в ячейке-счетчике (исходное состояние счетчика принято равным 8).

Первая последовательность						Вторая последовательность					
а	б	г	в	д	е	г	д	а	е	б	в
8	8	8	9	9	7	8	8	8	7	7	9

Смысл этих записей состоит в следующем. Если события а—е следуют не по порядку (а, б, в, г, д, е), а «переплетаются» между собой, то конечное состояние ячейки-счетчика (7 или 9) не соответствует начальному (8), хотя первый процессор прибавил, а второй отнял из текущего числа в ячейке-счетчике единицу.

Чтобы предотвратить такие конфликты, в ОЗУ выделяется некоторая ячейка-семафор. Нулевое состояние этой ячейки соответствует разрешению доступа к буферу и счетчику, ненулевое состояние соответствует запрету доступа. Прежде чем обратиться к буферу и счетчику, процессор (ЦП1 или ЦП2) выполняет команду «проверить и установить» (TEST & SET), в которой в качестве объекта проверки задана ячейка-семафор. Первая часть этой команды предписывает процессору проверить состояние ячейки-семафора, вторая — установить ненулевое состояние этой ячейки. На протяжении выполнения всей указанной команды магистраль занята процессором и никто из конкурентов не имеет права ею пользоваться (в магистрали выполняется «непрерываемая» процедура Чтение — модификация — запись, см. § 2.1).

Программа, выполняемая процессором, составлена так, что при запрещающем сигнале семафора проверки повторяются до тех пор, пока не будет обнаружен разрешающий сигнал. При обнаружении разрешающего сигнала, как уже отмечалось, процессор переводит семафор в запрещающее состояние и получает право доступа к буферу и счетчику. После окончания взаимодействия с буфером и коррекции содержимого счетчика процессор открывает семафор, т. е. засылает нулевой код в соответствующую ячейку ОЗУ. Таким образом, конфликты исключаются.

Однако эта система не лишена недостатков. Пока общий ресурс (буфер, счетчик) занят одним из процессоров, второй процессор, желающий воспользоваться тем же ресурсом, находится в состоянии ожидания, причем он циклически выполняет команду TEST & SET и тем самым загружает общую магистраль ненужными пересылками информации, мешая работать другим процессорам, подключенным к той же магистрали (на рис. 7.84 показаны только два процессора). Поэтому следовало бы построить систему таким образом, чтобы процессоры-конкуренты, желающие воспользоваться общим ресурсом, находились в режиме пассивного (а не активного) ожидания освобождения ресурса и не создавали на магистрали лишних пересылок информации. Иными словами, следовало бы ликвидировать «живую» очередь, как это принято во многих торговых предприятиях.

Для обеспечения пассивного ожидания предлагается использовать схему, показанную на рис. 7.85. По сравнению с предыдущей она содержит триггеры ТТ1 — $ТТ_N$, связанные входами S установки единицы и выходами Q с соответствующими процессорами ЦП1 — $ЦП_N$. Линия L выполняет функцию Монтажное ИЛИ по отношению к единичным сигналам, поступающим из процессоров. При наличии

единичного сигнала в линии L все триггеры переходят в нулевое состояние.

На начальном этапе работы с ресурсом (буфером и счетчиком) процессор, как и в предыдущей системе, выполняет команду TEST & SET и закрывает семафор, если он был открыт. Одновременно с выполнением этой команды процессор вырабатывает сигнал установки единицы соответствующего триггера TT_i. Процессоры-конкуренты также выполняют команды TEST & SET, но так как семафор уже закрыт, они не получают доступа к ресурсу. Выполнение команды TEST & SET процессорами-конкурентами также приводит к установке соответствующих триггеров TT_j в единичное состояние.

Смысл предлагаемой идеи заключается в том, что в результате установки в 1 триггеров TT_j соответствующие процессоры неспособны более выполнять команды TEST &SET и, дойдя до такой команды, процессоры приостанавливаются, не выходя на магистраль М (переходят в пассивное состояние).

Пока один из процессоров работает с ресурсом, конкуренты, однократно обратившись к семафору и выяснив, что он закрыт, пассивно ждут окончания использования ресурса. После освобождения ресурса процессор, владевший им, выдает единичный сигнал в линию L. Все триггеры, ранее установленные в 1, сбрасываются, и процессоры-конкуренты вновь вступают в борьбу за право пользования ресурсом. Таким образом, на этапе ожидания освобождения ресурса магистраль разгружается от «назойливых» обращений к ячейке-семафору со стороны процессоров-конкурентов.

Если в ОЗУ имеется несколько независимых групп семафор — счетчик — буфер, то и в этом случае система будет работать эффективно. При освобождении одного из ресурсов процессор, который его использовал, формирует сигнал на линии L, устанавливая в 0 все триггеры. При этом остальным процессорам не известно, какой именно из занятых ресурсов освободился. Все «заинтересованные» процессоры благодаря сбросу триггеров получают возможность однократной проверки нужного семафора. Если освободился именно тот ресурс, который нужен некоторому процессору и если ему повезло в борьбе с претендентами на использование этого ресурса, то процессор приступает к работе с ресурсом. Если же процессор обнаружил, что ресурс уже (все еще) занят, то он впадает в пассивное состояние ожидания благодаря установке в 1 соответствующего триггера TT_j, как было показано ранее.

Система обладает устойчивостью по отношению к сбоям в работе дополнительно введенной аппаратуры. Если в результате воздействия ложного сигнала один из ранее установленных в 1 триггеров TT_j перешел в нулевое состояние, то это приводит к однократному «лишнему» обращению со стороны соответствующего процессора к закрытому семафору, что не может привести к каким-либо нежелательным последствиям. При таком обращении триггер TT_j будет вновь установлен в единичное состояние, т. е. влияние сбоя компенсируется системой.

Если в результате появления ложного сигнала в линии L все ранее установленные в 1 триггеры оказались в нулевом состоянии, то это также не приведет к сбою в работе системы — процессоры, находившиеся в состоянии ожидания освобождения ресурса, выполнят по одной дополнительной команде TEST & SET и вновь перейдут в состояние ожидания, если ресурс еще занят.

Если триггер TT_k в результате воздействия ложного сигнала

устаивился в единичное состояние, то соответствующий процессор окажется неспособным работать с общим ресурсом, так как выполнение команды TEST & SET будет запрещено. Однако после того как ресурсом воспользуется один из конкурентов, триггер TT_k будет сброшен сигналом на линии L, т. е. система и в этой ситуации окажется устойчивой по отношению к сбою.

7.32. Повышение производительности двухмашинной системы с общим блоком памяти [141, 142]

На рис. 7.86, *а* показана традиционная схема связи двух микроЭВМ через общее ОЗУ. МикроЭВМ1 — центральная, микроЭВМ2 — подчиненная, ОЗУ доступно со стороны обеих микроЭВМ, арбитр А устраняет конфликты при одновременных обращениях к ОЗУ с обеих сторон.

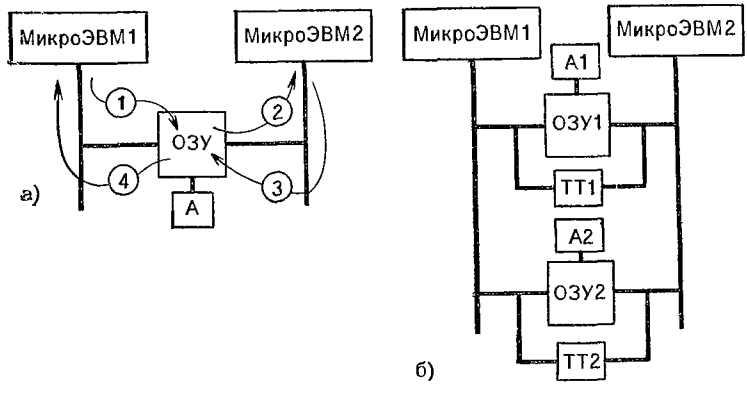


Рис. 7.86. Традиционная (*а*) и предлагаемая (*б*) схемы связи двух микроЭВМ через общую память

При работе системы центральная микроЭВМ выдает задание подчиненной, которая выполняет задание и готовит результат для использования центральной микроЭВМ. Последовательность событий такова.

1. МикроЭВМ1 записывает в ОЗУ информацию — задание для микроЭВМ2 (см. стрелку 1 на рис. 7.86, *а*).
2. МикроЭВМ2 считывает задание из ОЗУ (стрелка 2).
3. МикроЭВМ2 выполняет задание и записывает результаты в ОЗУ (стрелка 3).
4. МикроЭВМ1 считывает результаты из ОЗУ (стрелка 4).

При таком взаимодействии между микроЭВМ возможны потери времени из-за того, что события 1 и 2 или 3 и 4 могут «накладываться» друг на друга. Арбитр А исключает одновременные обращения к ОЗУ, поэтому одной микроЭВМ приходится ждать, пока другая не закончит обращение к общему ОЗУ.

В предлагаемом решении (рис. 7.86, *б*) потери времени при взаимодействии микроЭВМ уменьшены благодаря разделению общего ОЗУ на две (или более) части, при этом каждой части (ОЗУ1, ОЗУ2) поставлен в соответствие программно-доступный триггер (ТТ1, ТТ2), который выполняет функции «семафора».

МикроЭВМ1 при выдаче задания проверяет состояние триггера ТТ1. Если триггер установлен в 0, то это означает, что ОЗУ1 «свободно» и микроЭВМ1 записывает задание в это ОЗУ. Если в триггер ТТ1 записана 1, то это означает, что ОЗУ1 «занято» (интенсивно используется процессором микроЭВМ2) и микроЭВМ1 не обращается к нему, чтобы не мешать процессору микроЭВМ2, хотя обращения процессора микроЭВМ1 к ОЗУ1 не запрещены и арбитр этого ОЗУ при необходимости устранит возможные конфликты.

Если ОЗУ1 «занято», то микроЭВМ1 проверяет состояние триггера ТТ2 и определяет возможность записи задания в ОЗУ2. Если такая возможность имеется, то задание записывается в ОЗУ2, если нет, то микроЭВМ1 переключается на выполнение другой задачи (при наличии таковой) в ожидании освобождения ОЗУ1 или ОЗУ2, когда устанавливается в 0 триггер ТТ1 или ТТ2, что влечет прерывание микроЭВМ1 и возобновление попытки выдачи задания.

МикроЭВМ2 в интервалах между выполнениями задания просматривает определенные области ОЗУ1 и ОЗУ2, в которых может содержаться информация о новом задании. При обнаружении нового задания в ОЗУ1 (ОЗУ2) микроЭВМ2 программно устанавливает в 1 триггер ТТ1 (ТТ2) и приступает к выполнению задания с использованием ОЗУ1 (ОЗУ2). После выполнения задания микроЭВМ2 программно устанавливает в 0 триггер ТТ1 (ТТ2), «освобождая» ОЗУ1 (ОЗУ2).

7.33. Изменение параметров устройства, задаваемых микропереключателями, в отсутствие доступа к этим микропереключателям [143]

Предположим, что имеется герметически закрытый блок, внутри которого размещена микроЭВМ (рис. 7.87, *а*). Параметры микроЭВМ, такие, как собственный адрес в многомашинной системе, скорость передачи информации по каналу связи и др., задаются с помощью микропереключателей, размещенных внутри блока.

При выходе из строя микроЭВМ ее оперативно (на время ремонта) заменяют на исправную. Так как при такой замене нужно установить микропереключатели исправной микроЭВМ в положения, соответствующие параметрам неисправной микроЭВМ, пришлось бы разбирать (разгерметизировать) блок, чтобы получить доступ к микропереключателям, что в ряде случаев нежелательно или даже недопустимо.

Предлагается в подобных ситуациях электрически (а не «физически») отключать блок микропереключателей SW и подменять его набором перемычек на внешнем разъеме блока, см. рис. 7.87, *б*. (При штатной эксплуатации перемычек нет, они усложнили бы «кабельное хозяйство».)

Отключение микропереключателей производится установкой перемычки S1 на внешнем разъеме (рис. 7.88, *а*). Нулевые уровни напряжения в выбранных сигнальных линиях создаются установкой

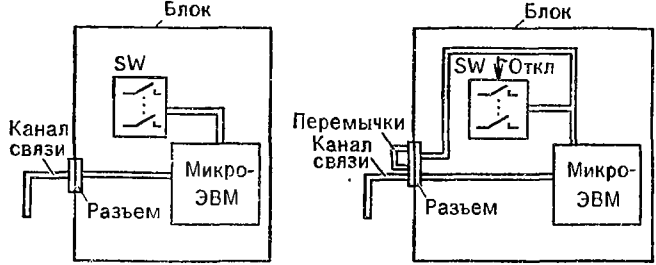


Рис. 7.87. Варианты конструкции системы, содержащей микроЭВМ, в которых блок микропереключателей SW недоступен для быстрого доступа.

В отличие от обычной схемы (*а*) предлагается в необходимых случаях отключать блок SW и подменять его перемычками, которые устанавливаются на внешнем разъеме блока (*б*)

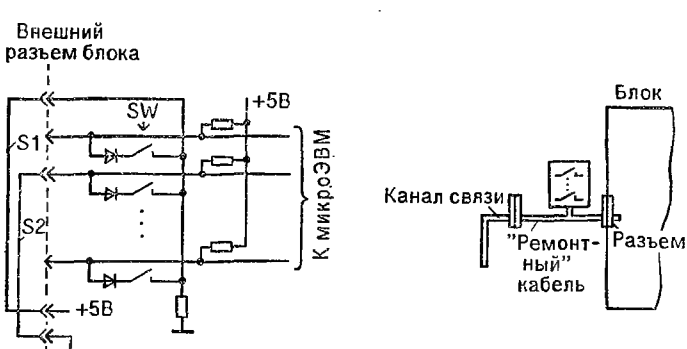


Рис. 7.88. Отключение блока микропереключателей и его подмена внешними перемычками (*а*) или внешними переключателями (*б*), установленными на кабельной вставке, используемой в период ремонта штатного блока

перемычек между этими линиями и шиной нулевого потенциала (см. перемычку S2).

Чтобы не устанавливать перемычки на разъеме, можно использовать «ремонтный» кабель — переходник с установленными на нем переключателями (аналогами перемычек S1, S2, рис. 7.88, *а*), обеспечивающими при необходимости отключение внутреннего блока SW и его подмену (рис. 7.88, *б*).

7.34. Подключение памяти на магнитных дисках к многопроцессорной системе [52]

На рис. 7.89 показаны примеры подключения дисковой памяти к многопроцессорной системе: рис. 7.89, *а* соответствует нерациональному подключеиию, а рис. 7.89, *б*, *в* — двум более приемлемым вариантам.

Рассмотрим рис. 7.89, *а*. Система содержит процессоры $ЦП_i$, соединенные с накопителями $Н_j$ на магнитных дисках, каждый из которых содержит пакет дисков D_j и контроллер $К_j$. Процессоры имеют двунаправленные внешние каналы связи для межпроцессорного взаимодействия и каналы для подключения периферийных устройств ввода-вывода, например терминала Т (с целью упрощения

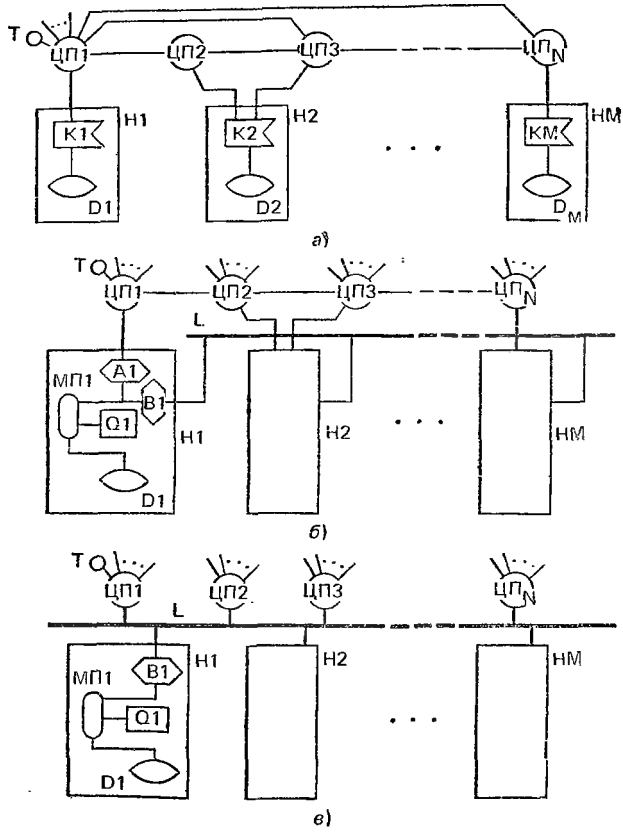


Рис. 7.89. Примеры подключения дисковой памяти к многопроцессорной системе

рисунка последняя группа каналов показана только для крайнего левого процессора).

Предположим, что оператор запросил с терминала (например, с алфавитно-цифрового дисплея) файл, который подлежит просмотру на экране и коррекции. Предположим далее, что этот файл расположен и пакете D2. Запрос с терминала поступает в процессор ЦП1, который, просматривая таблицу, определяет местоиахождение (на уровне накопителей) и размер искомого файла. Вся необходимая информация компонуется в сообщение, содержащее несколько элементов, таких, как имя файла, размер, тип операции (в данном случае — чтение) и т. п. Зная, что искомый файл расположен в накопителе Н2, процессор ЦП1 посылает сформированное сообщение в процессор ЦП3 через соответствующую межпроцессорную связь ЦП1—ЦП3. Процессор ЦП3 преобразует полученное сообщение на язык конкретных физических адресов и передает нужную информацию в коитроллер К2, который выполияет операцию поиска, считывания и передачи данных в ЦП3. Процессор ЦП3, используя межпроцессорную связь ЦП1—ЦП3, производит передачу файла в ЦП1, после чего ЦП1 выдает файл на экран дисплея (терминала Т). Как видим, процессор ЦП3 в данном случае является «лишним элементом» и выполияет функции буфера в ущерб полезной работе, на которую он ориентирован.

Схема, показанная на рис. 7.89, *б*, также содержит процессоры и накопители. Каждый накопитель включает интерфейсные блоки A_j и B_j для связи соответственно с процессором (или с несколькими процессорами) и с магистралью L, а также микропроцессор $МП_j$, память Q_j и пакет дисков. Рассмотрим выполнение той же задачи с использованием структуры, приведениой на рис. 7.89, *б*.

Запрос с терминала в этом случае в неизменном виде передается процессором ЦП1 через интерфейсный блок А1 в микропроцессор МП1, который под управлением собственной программы, хранимой в памяти Q1 или (и) в пакете дисков, опеделяет местоположение и размер искомого файла. Микропроцессор МП1, как и в предыдущем случае, формирует сообщение и пересылает его через интерфейсный блок В1, шину L и интерфейсный блок В2 в микропроцессор МП2 блока Н2. Микропроцессор МП2 выполияет те же функцин, что и процессор ЦП3 в схеме, показанной на рис. 7.89, *а*. Считанная с пакета D2 иифорнация пересылается по шине L в МП1. После этого микропроцессор МП1 пересылает файл в процессор ЦП1, который передает его на терминал.

В данном случае (рис. 7.89, *б*) значительная часть рутинной работы переложена на микропроцессоры, поэтому процессоры могут работать с большей производительностью. Кроме того, легко реализуется режим работы, при котором одйн файл размещается на нескольких пакетах из-за большой длины файла. Еще одно преимущестио — любой файл может быть логически доступен любому процессору.

Схема, показанная иа рис. 7.89, *в*, более проста и позволяет сокращать потери времени на передачу информации за счет непосредственной передачи файла из накопителя в процессор, связанный с запросившим обслуживанне терминалом. Недостаток (по сравнению с вариантом, показаниым на рис. 7.89, *б*) — невозможность распараллеливания процессов передачи информации между различными накопителямн и процессорами. «Свои» накопители приравнены теперь к «чужим».

7.35. Обмен информацией между процессорами через буфер типа FIFO [105]

Буфер типа FIFO обеспечивает, подобно конвейеру, последовательное накопление данных и их выдачу в порядке поступления. На рис. 7.90 показана схема связи двух процессоров через такой буфер.

Передача данных через буфер из процессора ЦП1 в процессор ЦП2 происходит при наличии единичных сигналов C1 и C2. В этом случае группы логических элементов И (D1, D2) готовы для передачи информации. Управляющая схема буфера вырабатывает сигнал

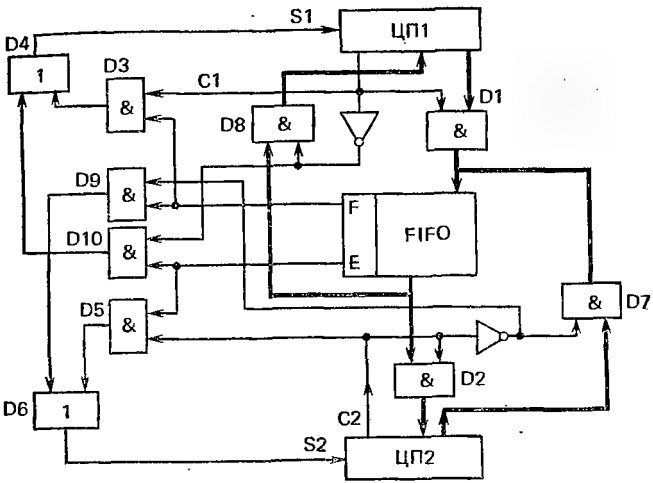


Рис. 7.90. Межпроцессорный обмен через буфер типа FIFO

F=1 при заполнении буфера до предела и сигнал E=1, если буфер пуст. Если буфер заполнен до предела, то срабатывают элементы И D3, ИЛИ D4 и в процессор ЦП1 поступает сигнал S1=1, который вызывает прекращение передачи информации из процессора ЦП1 в буфер. Если буфер пуст, то E=1, срабатывает цепь D5—D6 и в процессор ЦП2 поступает сигнал S2=1, препятствующий чтению информации процессором ЦП2 из буфера.

Передача данных через буфер из процессора ЦП2 в процессор ЦП1 происходит при C1=C2=0. В этом случае группы элементов И D7 и D8 подготовлены к передаче информации. При F=1 срабатывает цепь D9—D6 и вырабатывается сигнал S2=1, препятствующий дальнейшему пополнению буфера. При E=1 срабатывает цепь D10—D4 и вырабатывается сигнал S1=1, запрещающий чтение из буфера.

7.36. Многопроцессорная система, использующая табличную память для выбора страницы в главной памяти [73]

Система (рис. 7.91) содержит 12 исполнительных процессоров (ЦП1—ЦП12) и один «главный» процессор ЦП13, имеющий право изменять содержимое табличной памяти, которая по существу представляет собой 12 наборов 8-разрядных базовых регистров (для каждого процессора — свой набор). Главная память содержит 20 входиых адресных линий. Старшие 8 разрядов адреса передаются в главную память из табличной памяти по связям d, а младшие 12 разрядов — непосредственно из процессора, владеющего магистра-

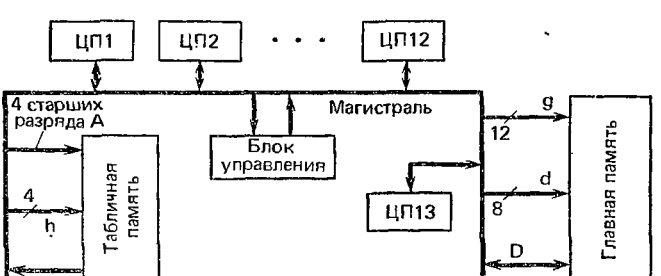


Рис. 7.91. Многопроцессорная система

лью. Блок управления выполняет функции арбитра.

Процессор, захвативший магистраль, выставляет на ней 16-разрядный адрес и собственный 4-разрядный адрес, который поступает в табличную память по связям h и выбирает нужную группу базовых регистров. Старшие 4 разряда 16-разрядного адреса не поступают в главную память, а служат для выбора одного из базовых регистров указаниой группы Содержимое выбранного базового регистра (8 бит) передается по связям d в качестве старших разрядов полного 20-разрядного адреса. Возможно каскадное включение табличных памятей для увеличения числа групп базовых регистров (например, для создания 64 групп, каждой из которых имеет право пользоваться любой процессор).

7.37. Уменьшение аппаратуры дополнительной памяти микроЭВМ [147, 187]

На рис. 7.92 показана микроЭВМ с подключенной к ней платой дополнительной памяти, которая содержит 4 микросхемы ОЗУ со структурой 2К×8 бит и микросхему дешифратора. Такое подключение неудобно, когда имеется несколько сменных плат с разными объемами памяти, так как в каждой такой плате необходимо устанавливать дешифратор. Предлагается перенести дешифратор на пла-

ту микроЭВМ. Это позволяет уменьшать затраты аппаратуры при построении ряда унифицированных плат расширения памяти микроЭВМ.

В примере, приведенном на рис. 7.93, плата дополнительной памяти также содержит 4 микросхемы ОЗУ со структурой 2К×8 бит, однако появляется возможность использования сменных плат с числом микросхем ОЗУ, увеличенным до 8 при подключении их входов выбора кристалла ВК к выходам С4—С7 дешифратора.

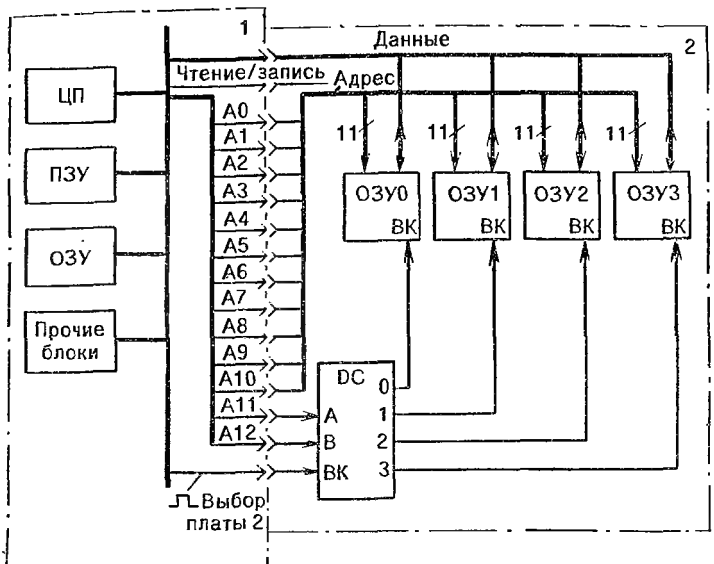


Рис. 7.92. Традиционная схема подключения к плате 1 микроЭВМ платы 2 дополнительной памяти.

Дешифратор установлен на плате 2

В примере, приведенном на рис. 7.94, дополнительное ОЗУ выполнено на основе 8 блоков памяти, каждый из которых содержит 8К ячеек; в примере, приведенном на рис. 7.95, использованы два дешифратора, один из которых (DC1) выбирает банк (64К ячеек), а другой (DC2) — блок памяти (16К ячеек) в пределах выбранного банка. Отметим, что предлагаемая система адресации не исключает возможности использования «обычных» плат дополнительной памяти, см. рис. 7.92. В этом случае дешифраторы, установленные на плате микроЭВМ, просто не используются, т. е. соответствующие контакты соединительного разъема остаются незадействованными.

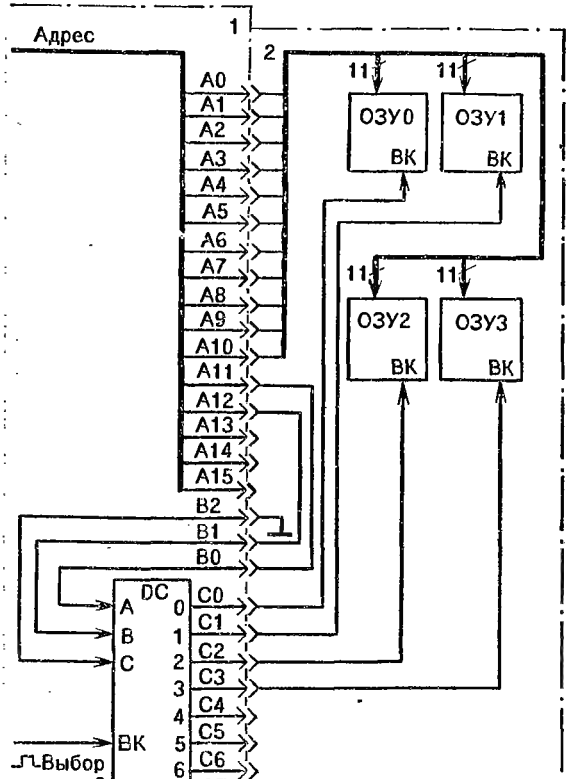


Рис. 7.93. Предлагаемая схема подключения к плате 1 микроЭВМ платы 2 дополнительной памяти.

Плата 2 не содержит «лишних» элементов, объем памяти может меняться при подключении дополнительных микросхем ОЗУ

7.38. Коррекция информации, хранимой в ПЗУ, с помощью дополнительной аппаратуры [149, 4]

При построении некоторых микропроцессорных устройств (кассовых аппаратов, устройств с речевым выводом информации и др.) необходима частая модернизация программ из-за изменяющихся внешних условий. Так как в подобных устройствах внешние накопители информации (например, жесткие или гибкие магнитные диски) обычно отсутствуют, программы хранятся во встроенном ПЗУ

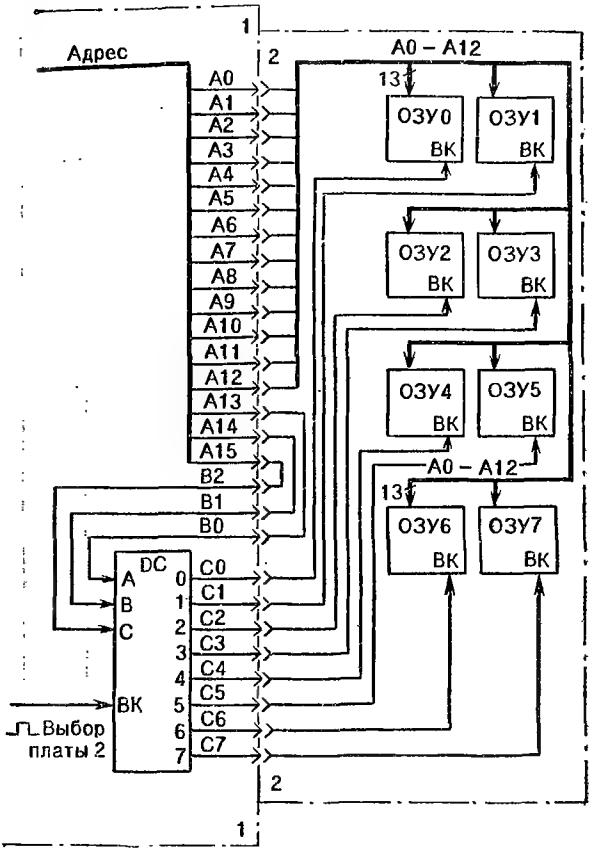


Рис. 7.94. Подключение к микроЭВМ дополнительной памяти 2, состоящей из 8 блоков по 8К ячеек

масочного типа. Чтобы изменить или ввести дополнительно хотя бы одну команду, потребовалось бы заказать микросхему ПЗУ с новой кодировкой, что слишком дорого.

В [149] предлагается ввести в состав микроЭВМ (рис. 7.96) дополнительную аппаратуру: ОЗУ1 — ОЗУ3 и контроллер бытового кассетного магнитофона (не показан) для ввода дополнительной информации о коррекции программы.

Каждой ячейке ПЗУ (рис. 7.97, *а*) соответствует один бит ОЗУ1 (признак М). Если М=0, то коррекции не требуется и центральный процессор ЦП, как обычно, работает по основной программе. Если

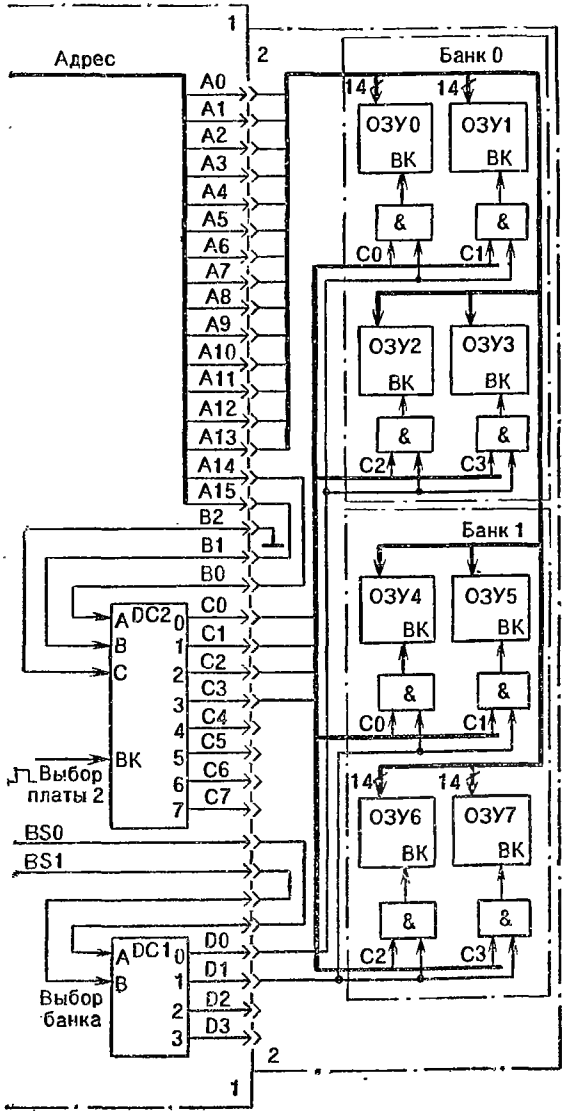


Рис. 7.95. Использование двух дешифраторов при подключении к микроЭВМ 1 дополнительной памяти 2

Рис. 7.96. Схема микроЭВМ с возможностью введения «вставок» в программы, хранимые в ПЗУ

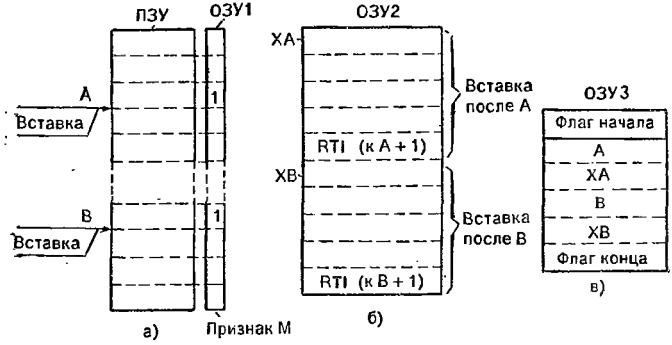


Рис. 7.97. Информация в ПЗУ и ОЗУ1 (*а*), ОЗУ2 (*б*) и ОЗУ3 (*в*).
После выполнения команды с адресом A (B) производится переход к программе-вставке с возвратом к основной программе

M=1, то процессор прерывает работу по основной программе, выполняет программу-вставку и продолжает работу, начиная с прерванного места. На рис. 7.97, *а* показаны две такие вставки, выполняемые после команд с адресами A и B. Каждая вставка заканчивается командой RTI — Выход из прерывания, отсылающей процессор к очередной команде с адресом A + 1 или B + 1.

В ОЗУ2 (рис. 7.97, *б*) хранятся программы-вставки. Адрес XA соответствует первой команде программы-вставки, выполняемой после команды с адресом A (рис. 7.97, *а*). Аналогично адрес XB соответствует началу второй программы-вставки. В ОЗУ3 (рис. 7.97, *в*) хранится таблица, помеченная признаками (флагами) начала и конца. В этой таблице содержится информация о том, какие адреса ПЗУ предшествуют вставкам и где находятся эти вставки. Например, информация, следующая после Флага начала, соответствует то-

му, что одна из вставок должна быть отработана после выполнения команды с адресом А (рис. 7.97, *а*), причем начальная команда этой вставки имеет адрес, равный ХА (рис. 7.97, *б*). Информация в ОЗУ3 необходима для начальной разметки ОЗУ1 и для перехода к нужной вставке при прерывании процессора.

Блок памяти ОЗУ4 (см. рис. 7.96) хранит «обычную» информацию, не имеющую отношения к предлагаемому решению. Блоки ОЗУ2

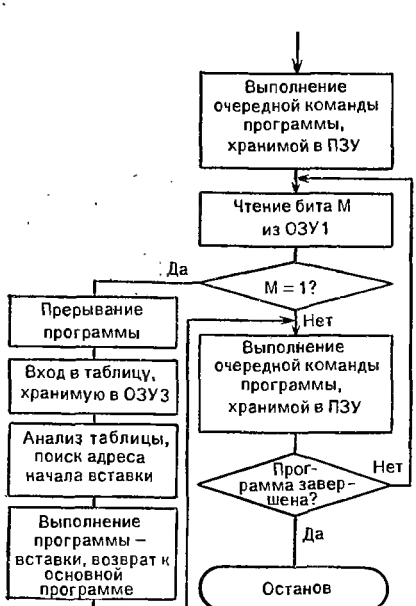


Рис. 7.98. Блок-схема алгоритма работы микроЭВМ (рис. 7.96)

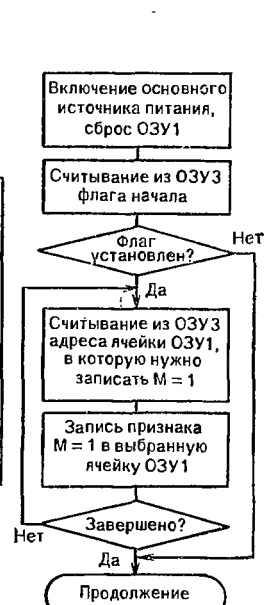


Рис. 7.99. Блок-схема алгоритма работы микроЭВМ после включения основного источника питания

и ОЗУ3 могут быть конструктивно объединены с блоком ОЗУ4 и показаны на рисунках раздельными лишь для удобства описания. Адреса ячеек ПЗУ и ОЗУ1 совпадают, адреса ячеек ОЗУ2—ОЗУ4 разные.

При работе процессора по основной программе одновременно со считыванием команды из постоянного запоминающего устройства из ОЗУ1 считывается признак М. Если М=1, то формируется сигнал INT прерывания процессора (см. рис. 7.96). Эти события соответствуют последовательной цепочке действий на блок-схеме алгоритма, приведенной на рис. 7.98. Прерывающая программа предписывает процессору обратиться в ОЗУ3 и проанализировать содержимое таб-

лицы (см. рис. 7.97, *в*) для отыскания кода А (В) и следующего за ним кода ХА (ХВ), после чего процессор отрабатывает нужную программу-вставку.

После включения основного источника питания (рис. 7.99) процессор выполняет программу пуска, хранимую в ПЗУ. На начальном этапе он не реагирует на случайные единичные сигналы прерывания, которые могут поступать из ОЗУ1 (ОЗУ1 не подключено к источнику резервного напряжения питания, и в нем может храниться случайная информация). В ходе выполнения программы пуска процессор последовательно записывает лог. 0 во все ячейки ОЗУ1, а затем разрешает реакцию на сигнал прерывания INT. (Установка в 0 ячеек ОЗУ1 может выполняться и аппаратно, но здесь этот вариант не рассматривается.)

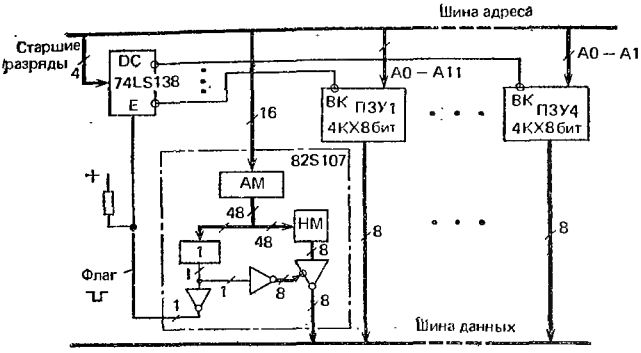


Рис. 7.100. Схема коррекции данных, хранимых в ПЗУ

Далее процессор проверяет, установлен ли Флаг начала в ОЗУ3. Если у микроЭВМ была «разумная» предыстория, то Флаг начала установлен (ОЗУ2—ОЗУ4 имеют резервную систему питания от батареи) и процессор, пользуясь информацией в ОЗУ3, размечает ОЗУ1, записывая лог. 1 по нужным адресам. Затем выполнение программы пуска продолжается (здесь не рассматривается как не представляющее интерес для описания предложенного решения). Если микроЭВМ включили «впервые», то Флаг начала не установлен, т. е. в соответствующей группе ячеек ОЗУ3, отведенных для его хранения, — случайная информация, с большой вероятностью не совпадающая с хранимым в ПЗУ эталоном. Если Флаг начала не установлен, то необходимо (если нужна коррекция) загрузить в ОЗУ2 и ОЗУ3 нужную информацию. Загрузка производится обычным путем, с использованием бытового кассетного магнитофона. После загрузки информации Флаг начала устанавливается в нужное состояние и процессор, вновь анализируя его, переходит к разметке ОЗУ1.

Отметим, что программа-вставка может быть составлена так, чтобы процессор не возвращался к точке прерывания, а уходил в любое другое место (при этом нужно учитывать, что содержимое регистра-указателя стека растет при каждом вхождении в про-

грамму-вставку и, следовательно, его нужно восстанавливать, чтобы не было переполнения).

В [4] решена более простая задача коррекции содержимого отдельных ячеек ПЗУ с использованием программируемой логической матрицы (ПЛМ) 82S107 (рис. 7.100). Адресная матрица (АМ) ПЛМ позволяет распознавать до 48 адресных кодов, в ответ на которые нужно произвести замену ошибочных данных на правильные, закодированные в накопительной матрице (НМ) ПЛМ.

При нормальной работе микроЭВМ ПЗУ1 — ПЗУ4 выдают данные в шину данных, когда это разрешено дешифратором. Однако когда появляется один из 48 закодированных в адресной матрице адресов, выставляется Флаг, запрещающий работу дешифратора, и, следовательно, ПЗУ, к которому обращался процессор (на рисунке он не показан), блокируется. Информация, соответствующая «ошибочной» ячейке ПЗУ, подменяется правильной информацией из накопительной матрицы НМ. Процессор при этом «не чувствует» подмены и работает как бы с новым (откорректированным) ПЗУ.

7.39. Простой интерфейс с совмещенной шиной адресов — данных [151]

Предлагаемая магистраль содержит, в частности, n-разрядную шину A/D для передачи адресов и данных, две управляющие линии для передачи сигналов C1 и C2 от активного устройства к пассивному и линию для передачи сигнала Ответ от пассивного устройства (остальные линии не представляют интереса). Рассмотрим режимы чтения и записи информации.

В режиме чтения (рис. 7.101, *а*) последовательность событий следующая.

Момент t_0. Захватив магистраль, активное устройство выдает в шину A/D адрес ячейки памяти (или любого адресуемого элемента), из которой нужно извлечь слово данных.

Момент t_1. Активное устройство формирует напряжения низкого уровня (лог. 0) на линиях C1 и C2. Интервал времени t_0—t_1 должен быть достаточным для установления правильного кода адреса в шине A/D с некоторым запасом, равным, например, 75 нс. Адресный код анализируется всеми пассивными устройствами, результаты анализа запоминаются ими в соответствующих триггерах, которые «защелкиваются» при переходе сигнала C1 (или C2) из 1 в 0. При правильном обращении адрес распознается как «свой» одним из пассивных устройств. (Если адрес не распознан ни одним из пассивных устройств, то возникает «зависание», которое устраняется активным устройством или схемой, подключенной к магистрали, см. гл. 1.)

Момент t_2. Пассивное устройство, ранее распознавшее на шине A/D свой адрес, формирует сигнал Ответ (лог. 0), который передается активному устройству.

Момент t_3. Активное устройство, узнав, что абонент подготовлен к обмену (Ответ = 0), освобождает шину A/D.

Момент t_4. Активное устройство формирует сигнал лог. 1 на линии C2. Это означает, что предстоит выполнить чтение информации из адресуемого элемента. Пассивное устройство приступает к выполнению операции чтения.

Момент t_5. Пассивное устройство выдает в шину A/D данные из адресуемого элемента.

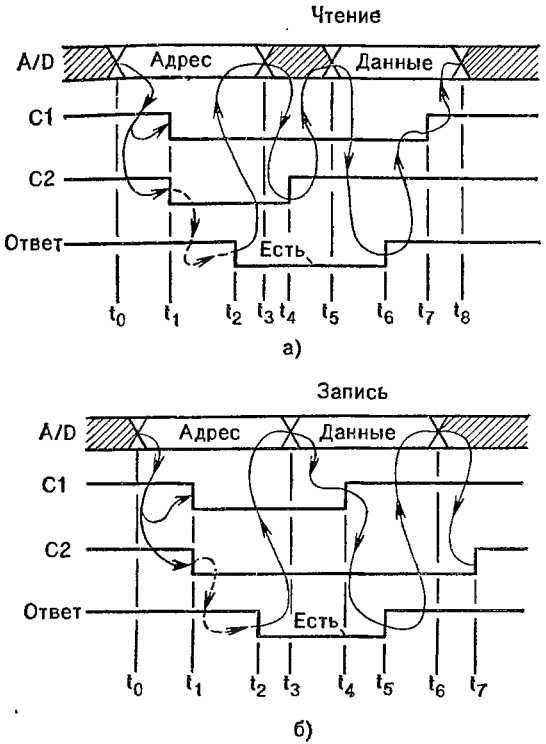


Рис. 7.101. Временные диаграммы чтения (*а*) и записи (*б*)

Момент t_6. Пассивное устройство подтверждает истинность данных, установленных в шине A/D, сигналом лог. 1 в линии Ответ. Задержка t_5—t_6 может компенсировать время переходных процессов в шине A/D; если эта задержка выбирается большей или равной нулю (как в интерфейсе Общая шина, см. гл. 2), то истинность данных по положительному фронту сигнала Ответ не гарантируется и активное устройство вынуждено само компенсировать указанное время.

Момент t_7. Активное устройство принимает данные и оповещает об этом пассивное устройство снятием сигнала лог. 0 с линии C1.

Момент t_8. Пассивное устройство снимает данные с шины A/D, освобождая ее.

В режиме записи (рис. 7.101, *б*) последовательность событий на начальном этапе такая же, как и при чтении. Отличия начинаются с момента t_3.

Момент t_3. Активное устройство снимает с шины A/D адрес и выдает в эту шину данные, которые нужно записать по указанному адресу.

Момент t_4. Активное устройство снимает сигнал лог. 0 с линии С1, сообщая пассивному устройству, что предстоит выполнить операцию записи. Интервал времени t_3—t_4 должен быть достаточным для установления данных в шине A/D с некоторым запасом.

Момент t_5. Пассивное устройство принимает данные и снимает сигнал лог. 0 с линии Ответ, сообщая активному устройству о том, что данные приняты.

Момент t_6. Активное устройство снимает данные с шины A/D.

Момент t_7. Активное устройство снимает сигнал лог. 0 с линии С2.

По сравнению с интерфейсом микроЭВМ типа «Электроника-60» в данном интерфейсе операции чтения и записи выполняются с использованием меньшего числа управляющих линий, а передача адреса производится асинхронно. В [151] приведены варианты построения многопроцессорных систем с данным интерфейсом.

7.40. Персональная микроЭВМ с возможностью использования процессоров разных типов [156]

Чтобы использовать широкий спектр программ, разработанных разными фирмами для процессоров разных типов, некоторые персональные микроЭВМ содержат несколько разных процессоров. Выбор нужного процессора производится в зависимости от выбранной оператором операционной системы или от других факторов (например, возможно переключение с одного процессора на другой при выполнении прикладной программы и т. п.). Предлагаемая схема позволяет подключать несколько процессоров к общей магистрали, содержащей шину адреса (A), данных (D) и управления (C).

МикроЭВМ (рис. 7.102) содержит процессоры ЦП1 и ЦП2, ПЗУ, ОЗУ и контроллеры $К_i$ внешних устройств; процессоры в данном примере имеют одинаковые интерфейсы. Для бесконфликтной работы процессоров в микроЭВМ введены триггеры ТТ1—ТТ3, элементы И1—И4 и дешифратор DC. Подключение к магистрали того или иного процессора определяется состоянием программно-доступного триггера ТТ3. Если этот триггер находится в состоянии Q=0, то разрешена работа процессора ЦП1; при Q=1 разрешена работа процессора ЦП2.

После включения напряжения питания формируется сигнал RESET, который устанавливает триггер ТТ3 в 0. Сигнал Q=0 проходит через элемент И3 на информационный вход триггера ТТ2 и по фронту сигнала F2 синхронизации процессора ЦП2 переписывается в триггер ТТ2. Сигнал HALT2=0 поддерживает процессор ЦП2 в заторможенном состоянии, при котором он отключен от магистрали. Это состояние подтверждается сигналом STATUS2=1, на выходах элементов И2 и И4 формируются сигналы лог. 1. По фронту сигнала F1 синхронизации процессора ЦП1 триггер ТТ1 устанавливается в 1 (HALT1=1), что соответствует разрешению работы процессора ЦП1. (Сигналы F1 и F2 могут формироваться двумя независимыми, не синхроиизированными между собой генераторами тактовой частоты.)

Далее процессор ЦП1, как обычно, работает с имеющимися ресурсами (памятью, контроллерами) до тех пор, пока ему не будет предписано передать «эстафету» процессору ЦП2. Для такой передачи процессор ЦП1 выполняет команду пересылки кода, содержа-

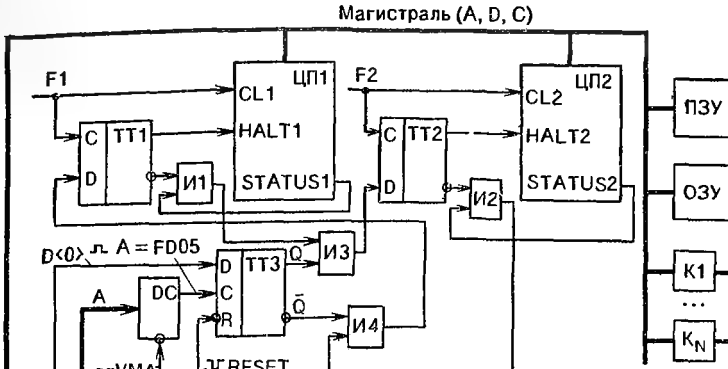


Рис. 7.102. «Двуликая» персональная мнкроЭВМ, работающая с двумя операционными системамн, соответствующими двум процессорам ЦП1 и ЦП2 разных типов

щего единицу в разряде D<0>, из внутреннего регистра общего назначения (РОН) в ячейку памяти с адресом, равным в данном примере FD05. Истинность адреса подтверждается сигналом VMA. В результате опознаиия адреса дешифратором DC формируется фронт сигиала на входе С триггера ТТ3, который устанавливается в 1.

Нулевой сигнал с выхода $\bar{Q}$ триггера ТТ3 проходит через элемент И4 и записывается в триггер ТТ1 (HALT1=0), процессор ЦП1 завершает выполнение текущей команды и отключается от магистрали (фактически процессор ЦП1 сам себя выключил). После отключения от магистрали процессора ЦП1 сигнал STATUS1=1 проходит через элементы И1, И3, триггер ТТ2 и включает в работу процессор ЦП2. Сигнал STATUS 2=0 поступает на элемент И2, в цепь формирования сигнала HALT1 и в дальнейшем исключает возможность преждевременного включения в работу процессора ЦП1 при обратной передаче эстафеты, когда триггер ТТ3 будет установлен в 0 процессором ЦП2.

Число процессоров может быть увеличено; при этом вместо триггера ТТ3 используется регистр и усложняются цепи взаимной блокировки процессоров.

7. 41. ПЗУ со страничной организацией [159]

Предлагаемая схема (рис. 7.103) позволяет подключить к 12-разрядной адресной шине микросхему ПЗУ с 13 адресными входамн. По существу используется метод базовых регистров (см. § 1.5), однако представляет иитерес схема загрузки этих регистров (RG0—RG2). Ее особенность в том, что она не подключена к шине данных и шине управления магистрали.

Старшие разряды адреса A10 и A11 делят адресное пространство (рис. 7.104, *а*) на 4 области по 1К ячеек. При обращении

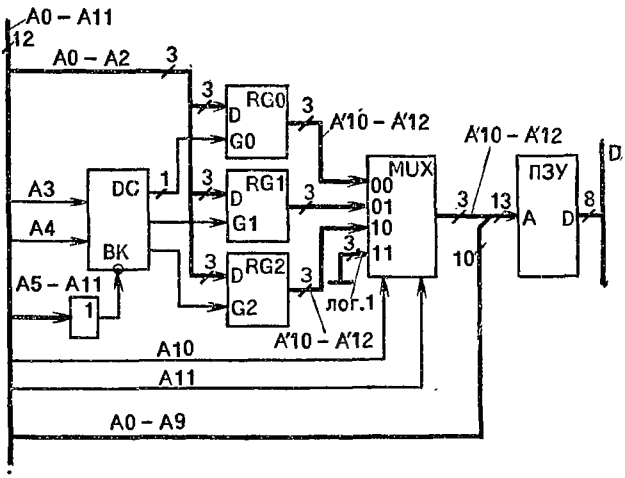


Рис. 7.103. Схема подключения ПЗУ, содержащего 2^{13} ячеек, к 12-разрядной адресной шине

в область 00 мультиплексор MUX передает на адресные входы ПЗУ три старших разряда адреса из регистра RG0, в результате выбирается одна из 8 страниц ПЗУ, содержащая 1К 8-разрядных ячеек. Аналогично при обращении в области 01 и 10 старшие разряды адреса поступают в ПЗУ из регистров RG1 и RG2. В зависимости от содержимого выбранного регистра производится обращение к одной из 8 страниц ПЗУ.

При обращении в область 11 адресного пространства всегда производится выбор последней страницы ПЗУ независимо от содержимого регистров RG0—RG2. Это связано с тем, что нижняя группа входов мультиплексора подключена к шине нулевого потенциала (лог. 1). Выбор последней страницы ПЗУ может производиться и через регистр RG0, RG1 или RG2, если в нем присутствует код 111_2.

Особенность данной схемы состоит в том, что загрузка регистров RG0—RG2 производится автоматически при обращении к 24 последним ячейкам адресного пространства. При обращении по адресу FE8 (см. рис. 7.104, *б*) в регистр RG0 загружается нулевой код, при обращении по последующим семи адресам в регистр RG0 загружаются коды 1, 2, ..., 7_8. Аналогично, выбрав нужные адреса, можно загрузить любые 3-разрядные коды в регистры RG1 и RG2.

Таким образом, при обращении в область 11 адресного пространства можно получить доступ к 1К—24 ячейкам последней страницы ПЗУ. Однако при обращении к последним 24 ячейкам в качестве «побочного эффекта» производится загрузка регистров RG0—RG2 соответствующей информацией. Поэтому можно считать, что обращение к последним 24 ячейкам ПЗУ через область 11 запрещено. При обращении к этим ячейкам через области 00, 01 и 10 «побочных эффектов» не возникает, т. е. последняя страница ПЗУ оказывается

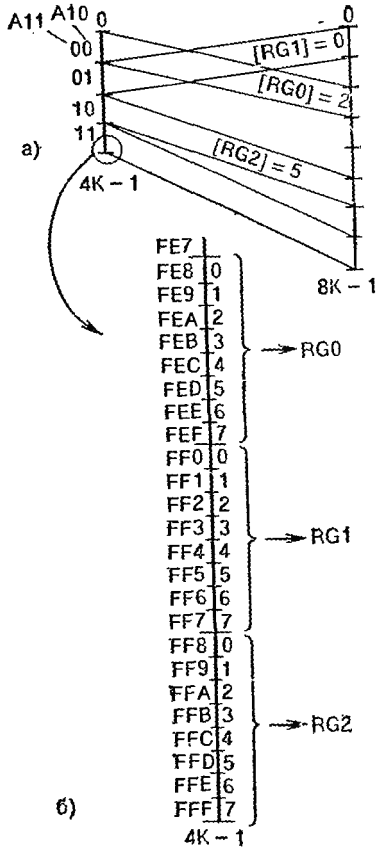


Рис. 7.104. Преобразование адресного пространства *(а)* и развернутое изображение *(б)* его нижней части, используемой при загрузке регистров RG0—RG2

в полной мере доступной для считывания информации в шину данных D.

В исходном (пассивном) состоянии во всех двенадцати линиях адресной шины присутствуют напряжения H высокого уровня (лог. 0). При загрузке регистра (RG0—RG2) в разрядах A5—A11 адресной шины устанавливаются напряжения L низкого уровня, на выходе элемента ИЛИ формируется сигнал (L-уровень), отпирающий дешифратор DC. В зависимости от комбинации сигналов в разрядах A3 и A4 (01, 10 или 11) на одном из выходов дешифратора формируется H-уровень — сигнал выбора регистра RG0—RG2. Этот сигнал поступает на управляющий вход G_i выбранного регистра RG_i и вызывает прием информации с линий A0—A2 адресной шины. Таким образом при загрузке регистра RG_i три младших разряда адресной шины используются не по прямому назначению и служат для передачи данных в регистр.

Регистр RG_i (рис. 7.105) содержит две последовательно соединенные группы RS-триггеров — входную и выходную. В исходном состоянии (рис. 7.106, *а*, период времени до момента t_0) A_i=H, G_i=L, B=C=H, входная группа триггеров транслирует сигналы H-уровня на выходы Q_i, выходная группа триггеров закрыта по входам сигналом E=L, поэтому на выходах A′10—A′12 присутствует старая информация.

Из-за нестрогой одновременности поступления сигналов по адресной шине, а также с учетом задержек их прохождения через элемент ИЛИ и дешифратор в период времени t_0—t_1 возможны переходные процессы (заштрихованы), по окончании которых в выбранном регистре G_i=H, B=L, Q_i=A_i, информация на выходах A′10—A′12 не меняется. В период времени t_2—t_3 могут наблюдаться переходные процессы на выходах A′10—A′12. При этом триггеры входной группы однократно или многократно (из-за возможной неста-

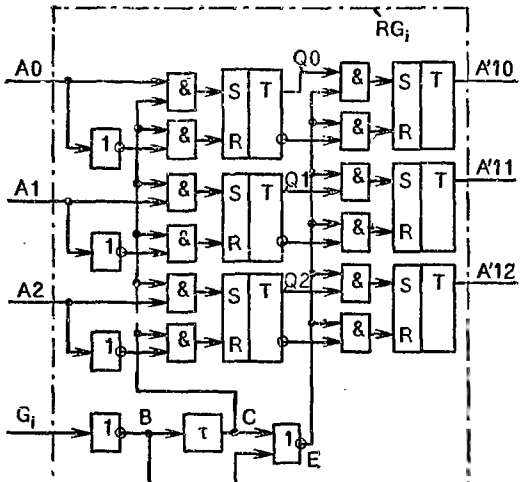


Рис. 7.105. Структура регистра RG_i.

Сигнал G_i формируется из адресных сигналов A3—A11, однако гонок не возникает

бильности сигнала C) «защелкиваются» по входам, что, однако, не нарушает стабильности информации в этих триггерах. После момента t_3 на выходах A'10—A'12 устанавливается новая информация, триггеры выходной группы открыты по входам, триггеры входной группы отключены от входов A0—A2.

В период времени t_4—t_5 происходит однократная или многократная (из-за нестабильности сигнала E) фиксация новой информации в триггерах выходной группы, при этом нестабильность сигнала E не может привести к искажениям информации. В период времени t_6—t_7 триггеры входной группы переходят в режим передачи входной информации на выходы, регистр переходит в исходное состояние.

Схема гарантирует сохранность информации в невыбранных регистрах (рис. 7.106, *б*). Переходные процессы в точке C (период времени t_2—t_3) не меняют состояния триггеров входной группы. Благодаря разнесению во времени периодов нестабильности сигналов в точках B и C, сигнал E=L, триггеры выходной группы постоянно закрыты по входам и хранят старую информацию.

Таким образом, данная схема «самоуправляема» и обеспечивает запись информации в регистры RG0—RG2 без использования внешних синхросигналов.

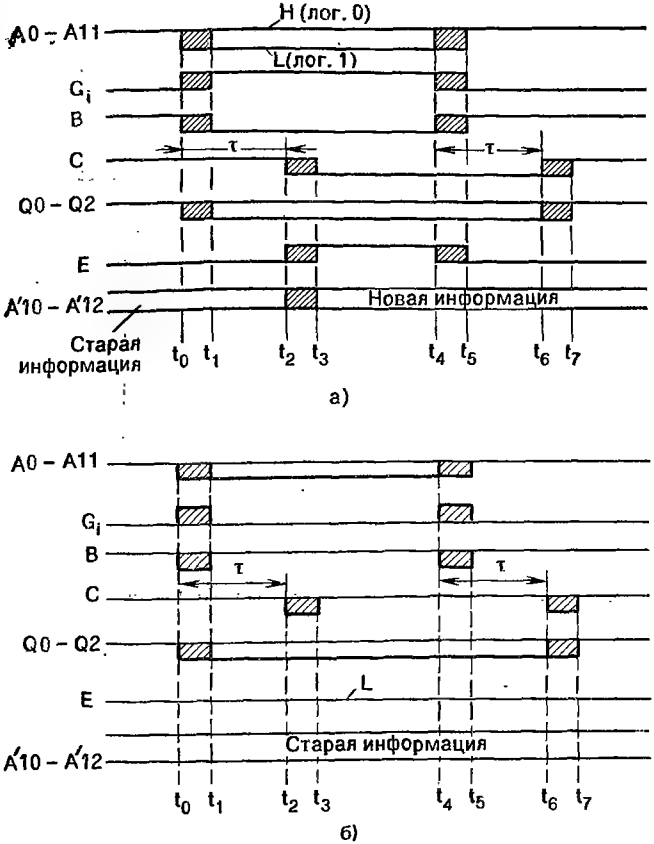


Рис. 7.106. Работа выбранного (а) и невыбранного (б) регистра RG_i.
Переходные процессы (заштрихованы) не создают опасности записи ложной информации в регистры

7.42. Расширение памяти микроЭВМ подключением дополнительных блоков разного объема [170, 180]

Для расширения памяти микроЭВМ обычно используют дополнительные платы ОЗУ или ПЗУ, которые устанавливаются в незадействованные разъемы магистрали. Объем памяти на плате может быть разным, например 8, 16, 32, 64, 128, 256, 512, 1024К байт. При установке дополнительных плат памяти разного объема возникает задача их «плотного размещения» в адресном пространстве микроЭВМ.

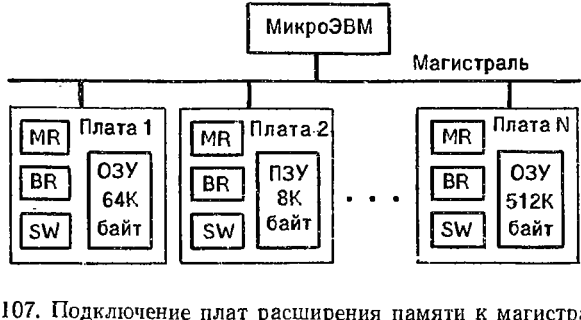


Рис. 7.107. Подключение плат расширения памяти к магистрали микроЭВМ

Предлагаемое решение этой задачи (рис. 7.107) позволяет пользователю не заботиться о порядке установки в разъемы дополнительных плат и задании начальных адресов памяти для каждой платы. Операционная система без участия человека-оператора определяет состав дополнительных плат и компонует единый дополнительный блок памяти, размещая его в определенной области адресного пространства микроЭВМ.

После включения напряжения питания регистры BR устанавливаются в некоторое исходное состояние, так что все дополнительные блоки памяти автоматически размещаются в некоторой (возможно, общей для всех блоков) «далекой» области свободного адресного пространства и «не мешают» работе микроЭВМ. Процессор микроЭВМ последовательно опрашивает разъемы, в которые могут быть установлены дополнительные платы, и определяет, какие из этих разъемов заняты, а какие нет. В каждой плате имеется трехразрядный переключатель SW (или три паяные перемычки), который задает объем памяти на плате. Например, код 000_2 соответствует объему памяти 8К байт, код 001—16К байт и т. д. МикроЭВМ опрашивает переключатели SW и «узнаёт» о том, какой объем адресного пространства следует выделить каждой плате. В зависимости от необходимого каждой плате объема адресного пространства микроЭВМ записывает в программно-доступные регистры маски MR коды, представленные в табл. 7.2. Адрес регистра MR (и соответствующего регистра BR) зависит от того, в какой разъем установлена плата.

В данном примере в регистры MR плат 1, 2 и N записываются коды 11111000, 11111111 и 11000000. Далее микроЭВМ определяет, является ли память в каждой плате постоянной (доступной только по чтению) или оперативной (доступной по чтению и по записи). Для этого каждый проверяемый блок памяти временно переносится в определенную незанятую область адресного пространства путем загрузки в соответствующий регистр BR нужных кодов. После тестирования блока определяется его принадлежность ОЗУ или ПЗУ и проверяется работоспособность (ПЗУ могут в определенных ячейках содержать контрольные суммы, которые сверяются с вычисленными).

Получив полную информацию о составе плат, микроЭВМ производит их размещение в адресном пространстве: ОЗУ собираются в

одну группу, ПЗУ — в другую. При этом быстродействующие статистические ОЗУ малого объема могут компоноваться отдельно от динамических («медлениых») ОЗУ большого объема, что позволит в дальнейшем повысить эффективность использования памяти в целом. Порядок размещения ПЗУ в адресном пространстве выбирается в соответствии с дополнительными признаками — содержимым определенных ячеек этих ПЗУ и т. п.

Таблица 7.2

Объем памяти на плате, К байт	Содержимое регистра MR							
8	1	1	1	1	1	1	1	1
16	1	1	1	1	1	1	1	0
32	1	1	1	1	1	1	0	0
64	1	1	1	1	1	0	0	0
128	1	1	1	1	0	0	0	0
256	1	1	1	0	0	0	0	0
512	1	1	0	0	0	0	0	0
1024	1	0	0	0	0	0	0	0

Схема управления адресацией платы (рис. 7.108) содержит регистр маски MR, регистр BR дополнительного кода базового адреса, сумматор, группу элементов И и выходной элемент ИЛИ. Схема представляет собой программируемый дешифратор старших разрядов адреса для выявления адресов, относящихся к ячейкам памяти, размещеиным в данной плате. При выявлении таких адресов на выходе элемента ИЛИ формируется сигнал лог. 0, при обращении по «чу-

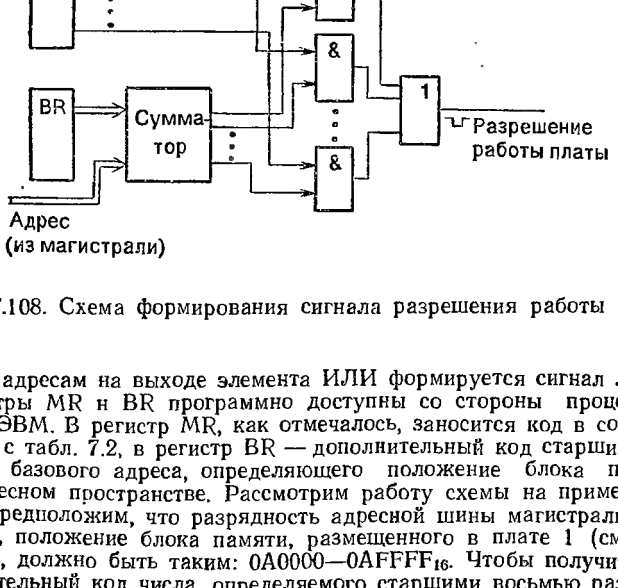


Рис. 7.108. Схема формирования сигнала разрешения работы платы

жим» адресам на выходе элемента ИЛИ формируется сигнал лог. 1. Регистры MR и BR программно доступны со стороны процессора микроЭВМ. В регистр MR, как отмечалось, заносится код в соответствии с табл. 7.2, в регистр BR — дополнительный код старших разрядов базового адреса, определяющего положение блока памяти в адресном пространстве. Рассмотрим работу схемы на примере.

Предположим, что разрядность адресной шины магистрали равна 21, положение блока памяти, размещенного в плате 1 (см. рис. 7.107), должно быть таким: 0A0000—0AFFFF$_{16}$. Чтобы получить дополнительный код числа, определяемого старшими восьмью разрядами 01010000 базового адреса 0A0000, эти разряды инвертируются

и к полученному числу арифметически прибавляется единица младшего разряда (эти операции выполняются процессором микроЭВМ, полученный результат записывается в регистр BR):

```
01010000 — Старшие разряды базового адреса
10101111 — Инвертированные разряды
+
       1 — Прибавление единицы
10110000 — Дополнительный код старших разрядов базового адреса, загружаемый в регистр BR платы 1
```

При обращении микроЭВМ к ячейке памяти с адресом 0AXXXX (X — произвольное число из диапазона 0—F), принадлежащим области адресов 0A0000 — 0AFFFF, сумматор (см. рис. 7.108) складывает два восьмиразрядных кода, причем перенос из старшего разряда теряется:

```
10110000 — Содержимое регистра BR
+
01010XXX — Адрес из магистрали (X—0 или 1)
00000XXX — Код на выходе сумматора
```

Полученный код поразрядно логически умножается на код 11111000 в регистре маски MR с помощью элементов И:

```
00000XXX — Код на выходе сумматора
&
11111000 — Код в регистре маски MR
00000000 — Код на входах элемента ИЛИ
```

Так как на всех входах элемента ИЛИ присутствуют нули, на его выходе формируется лог. 0 — признак попадания адреса в диапазон адресов, выделенный для платы 1. Из приведенного примера следует, что нули в пяти старших разрядах результата суммирования кодов формируются в том и только в том случае, когда поступивший из магистрали адрес содержит в старших разрядах комбинацию цифр 0A. Любая отличная от этой комбинация цифр дает результат суммирования, содержащий единицу хотя бы в одном из пяти старших разрядов. При этом срабатывает один или несколько элементов И, на входы элемента ИЛИ поступает одна или несколько единиц, на выходе этого элемента формируется сигнал лог. 1 — признак запрета работы платы 1.

Переключатели SW можно исключить из платы, при этом объем имеющейся на плате памяти определяется процессором микроЭВМ после проведения серии «программных» экспериментов, в каждом из которых проверяется очередная гипотеза о существовании на плате 8K, 16K, 32K и т. д. ячеек. Вместо регистра MR можно использовать набор паяных перемычек, жестко задающих объем памяти согласно табл. 7.2. Если обеспечить программное считывание кода, задаваемого перемычками, то можно получить информацию об объеме памяти на плате, переключатели SW окажутся ненужными. Возможны и иные изменения и улучшения. Например, неисправные блоки памяти могут программно «выбрасываться» в неиспользуемую часть адресного пространства, а оставшиеся исправные — сдвигаться по адресам вплотную друг к другу, чтобы между ними не оставалось незанятых областей.

7.43. Программируемый адресный дешифратор [171]

При построении микропроцессорных устройств с перестраиваемой архитектурой, например работающих с разными операционными системами, может возникнуть необходимость перемещения адресов различных элементов в ту или иную область адресного пространства. «Настройка» адресов может производиться центральным процессором после включения напряжения питания устройства и распознания типа используемой операционной системы.

Предлагаемая схема (рис. 7.109) позволяет получать две группы дешифрируемых адресов, которые можно перемещать в адресном пространстве независимо друг от друга. Разряды 0—3 и 4—7

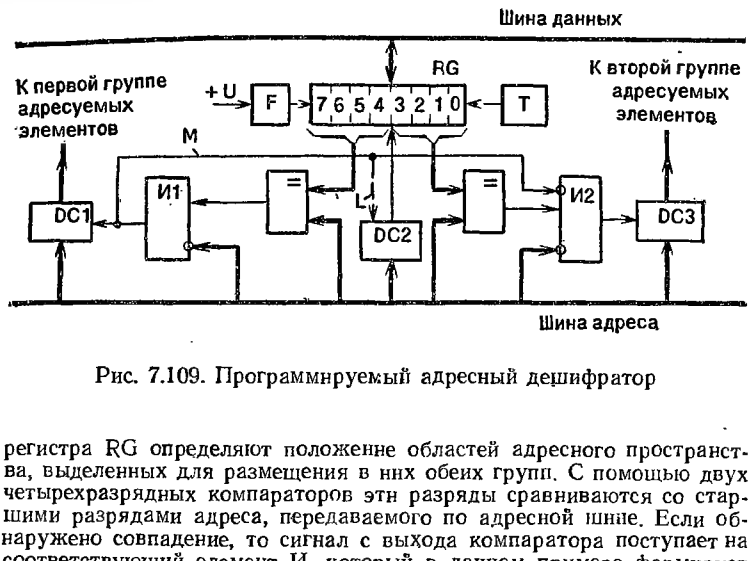


Рис. 7.109. Программируемый адресный дешифратор

регистра RG определяют положение областей адресного пространства, выделенных для размещения в них обеих групп. С помощью двух четырехразрядных компараторов эти разряды сравниваются со старшими разрядами адреса, передаваемого по адресной шине. Если обнаружено совпадение, то сигнал с выхода компаратора поступает на соответствующий элемент И, который в данном примере формирует сигнал лог. 1, если в разрядах адреса, поступающих на его инверсные входы, присутствуют лог. 0. При этом открывается дешифратор DC1 или DC3, анализирующий разряды адреса, поступающие на его входы; с выходов дешифраторов снимаются сигналы выбора адресуемых элементов — регистров устройств ввода-вывода, блоков памяти и т. п.

Если в разрядах 0—3 и 4—7 регистра RG записаны одинаковые коды, то оба компаратора при обнаружении соответствующего адреса на адресной шине формируют на выходах сигналы лог. 1. Если при этом сработал элемент И1, то сигнал M=1 с его выхода запирает элемент И2. Таким образом, если первая и вторая группа адресов частично накладываются друг на друга, то предпочтение отдается первой группе и конфликтов не возникает. Например, регистры устройств ввода-вывода (первая группа) могут «вытеснить» в адресном пространстве соответствующие ячейки ОЗУ (вторая группа).

Регистр RG программно доступен со стороны центрального процессора. Его адрес распознаётся дешифратором DC2. При наличии связи L, показанной на рис. 7.109 штриховой линией, разрешение работы дешифратора DC2 производится только при M=1, т. е. при попадании адреса в диапазон адресов, дешифрируемых элементом И1. В этом случае информационные входы дешифратора DC2 подключаются не ко всем разрядным линиям адресной шины. В исходном состоянии, после включения напряжения питания, в регистре RG схемно устанавливается определенный код. Разряды 4—7 этого кода определяют, в частности, адрес регистра RG. После обращения центрального процессора по этому адресу и смены информации в разрядах 4—7 регистра RG его адрес перемещается вместе с адресами первой группы элементов.

Чтобы защитить регистр RG от несанкционированной смены информации, например в результате беспорядочного «блуждания» процессора по памяти из-за ошибки программиста, можно использовать одновибратор F и триггер T. При включении напряжения питания одновибратор вырабатывает импульс длительностью, например, 100 мкс. В течение этого времени доступ к регистру RG со стороны центрального процессора разрешен, а в дальнейшем регистр можно только прочитать. Таким образом, запись информации в регистр RG разрешена только при выполнении начального участка программы инициализации устройства. Триггер T разрешает произвести только одну запись данных в регистр RG, затем этот триггер устанавливается в запрещающее состояние и находится в нем вплоть до момента выключения напряжения питания.

7.44. Взаимодействие между центральной и периферийными микроЭВМ с использованием общего ОЗУ [174]

В многомашинной системе (рис. 7.110) микроЭВМ16 — центральная, микроЭВМ 0 — 15 — периферийные. Центральная микроЭВМ по мере необходимости устанавливает связь с одной из периферийных. Для обмена данными используется общее ОЗУ.

Центральная микроЭВМ имеет магистральные входы-выходы M и выходной порт (программно-доступный выходной регистр). Каждая периферийная микроЭВМ также имеет магистральные входы-выходы M и входной порт (группу программно-доступных входных линий). Входы-выходы M служат для передачи сигналов адреса, данных, записи (WR, WR′) и подтверждения истинности адреса (VMA, VMA′). Эти сигналы формируются центральной или периферийной микроЭВМ при необходимости обращения к общему ОЗУ; при работе микроЭВМ с внутренними ОЗУ, ПЗУ и другими элементами (не показаны) входы-выходы M находятся в пассивном состоянии.

С помощью мультиплексоров MS в зависимости от сигнала H общее ОЗУ подключается либо к магистрали центральной, либо к магистрали периферийных микроЭВМ, причем старшие разряды адреса ОЗУ в последней ситуации принимаются из выходного порта микроЭВМ16. Так как шина данных ОЗУ двунаправленная, соответствующий мультиплексор обеспечивает двустороннюю передачу данных. Направление передачи зависит от значения сигнала WR″:

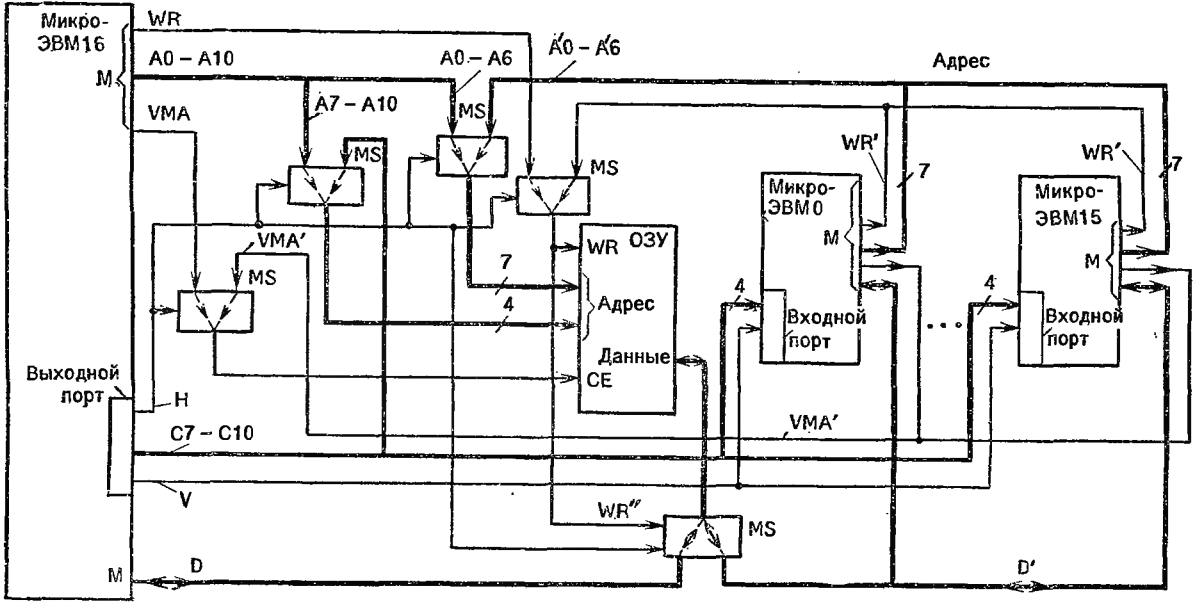


Рис. 7.110. Многомашинная система с общим ОЗУ

при записи данные поступают в ОЗУ, при чтении — к микроЭВМ.

Четыре старших разряда адреса ОЗУ (2К×16 бит) делят его на 16 страниц, по 128 ячеек на каждой. Каждая страница предназначена для обмена данными между центральной и одной из периферийных микроЭВМ. Центральная микроЭВМ имеет доступ ко всем страницам, периферийная — только к «своей» странице, когда мультиплексоры подключают общее ОЗУ к магистрали периферийных микроЭВМ.

Система работает следующим образом. В исходном состоянии H=0, V=0, в линиях C7—C10 выходного порта присутствует произвольный код, общее ОЗУ подключено через мультиплексоры MS к центральной микроЭВМ. Центральная микроЭВМ работает по программе, хранимой во внутреннем ПЗУ или ОЗУ, и при необходимости обращается к общему ОЗУ для считывания или записи кодов. Периферийные микроЭВМ, работая по внутренним программам, периодически опрашивают свои входные порты, дожидаясь момента появления сигнала V=1. Общая магистраль периферийных микроЭВМ находится в пассивном состоянии.

Если центральная микроЭВМ желает передать в одну из периферийных некоторую информацию (задание), то эта информация помещается ею на соответствующую страницу общего ОЗУ. После этого центральная микроЭВМ устанавливает в линиях C7—C10 код (в диапазоне 0000—1111_2), соответствующий выбранной периферийной микроЭВМ, затем устанавливает сигнал H=1, подключающий общее ОЗУ к магистрали периферийных микроЭВМ, и, наконец, формирует сигнал V=1. Каждая периферийная микроЭВМ, обнаружив при очередном опросе входного порта, что V=1, считывает и анализирует код, установленный в линиях C7—C10. Одна из периферийных микроЭВМ определяет, что этот код совпадает с ее собственным четырехразрядным номером, например заданным микропереключателями, установленными на плате. Остальные периферийные микроЭВМ продолжают находиться в режиме ожидания следующего сеанса связи.

Выбранная периферийная микроЭВМ имеет внешнее адресное пространство объемом $2^7=128$ ячеек, т. е. способна работать только с одной страницей. Номер страницы постоянно передается в ОЗУ через мультиплексор четырех старших разрядов адреса с линий C7—C10 выходного порта микроЭВМ16, т. е. схема обеспечивает точное соответствие между номерами периферийных микроЭВМ и номерами выделенных им страниц общего ОЗУ. Обращаясь к общему ОЗУ, выбранная микроЭВМ автоматически попадает в выделенную ей страницу, считывает приготовленное для нее задание и помещает в это ОЗУ результаты выполнения предыдущего задания, если система работает в установившемся режиме и имеет предысторию.

Через некоторое время, достаточное для завершения обмена информацией между выбранной периферийной микроЭВМ и общим ОЗУ, центральная микроЭВМ устанавливает в 0 сигнал V, затем сигнал H, подключая общее ОЗУ к своей магистрали. После этого центральная микроЭВМ считывает результаты работы выбранной периферийной микроЭВМ из общего ОЗУ и использует их для вычислений. Сеанс связи с любой другой периферийной микроЭВМ протекает аналогично. Непосредственная передача информации из одной периферийной микроЭВМ в другую не предусмотрена.

Возможны иные способы осуществления взаимодействия между

центральной и периферийной микроЭВМ, например с использованием линий прерывания. Можно уменьшить до двух число разрядов выходного порта центральной микроЭВМ, если линии С7—С10 подключить к выходам внешнего двоичного четырехразрядного счетчика, рис. 7.111. При каждом переходе сигнала Н из 0 в 1 к содержимому счетчика СТ прибавляется единица, т. е. осуществляется поочередная связь центральной микроЭВМ со всеми периферийными. Чтобы определить текущее состояние счетчика, центральная микроЭВМ записывает в каждую страницу задание, предписывающее периферийной микроЭВМ переслать некоторый ненулевой код, например свой номер, увеличенный на единицу, в определенную (предва-

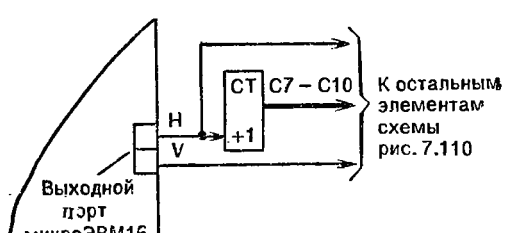


Рис. 7.111. Доработка схемы рис. 7.110 для уменьшения числа линий выходного порта микроЭВМ16

рительно очищенную центральной микроЭВМ) семафорную ячейку на странице. Затем, как было описано, общее ОЗУ сигналами Н=1 и V=1 предоставляется в распоряжение одной из периферийных микроЭВМ, а именно той, которая распознаёт на линиях С7—С10 свой номер. Эта микроЭВМ выполняет задание, а остальные остаются в пассивном состоянии.

Центральная микроЭВМ после паузы, гарантирующей выполнение задания периферийной микроЭВМ, возвращает себе общее ОЗУ и анализирует содержимое семафорных ячеек. В одной из них должен присутствовать ненулевой код. Уменьшив его на единицу, центральная микроЭВМ определит номер периферийной микроЭВМ, выполнившей задание. (Этот же результат содержится в четырех старших разрядах адреса ненулевой семафорной ячейки общего ОЗУ). Этот номер совпадает с содержимым счетчика СТ.

Чтобы перейти от текущего состояния счетчика к любому другому, центральная микроЭВМ программно формирует нужное число импульсов Н, сохраняя условие V=0. При этом происходит ряд «холостых» переключений мультиплексоров MS, что, однако, не приводит к каким-либо нежелательным последствиям.

Возможны и иные изменения схемы. Например, мультиплексоры можно исключить, а соответствующие линии объединить, если допустимо их прямое электрическое соединение в «общие шины», и т. п.

7.45. Повышение быстродействия кэш-памяти [183]

Кэш-память (от англ. cache — что-либо припрятанное) представляет собой быстродействующее статическое ОЗУ небольшого объема (например, 256 ячеек), в котором по мере работы процессора «припрятывается» наиболее актуальная информация (рис. 7.112). Эта информация первоначально хранится в динамическом ОЗУ, быстродействие которого примерно в 10 раз ниже статического.

При считывании информации из динамического ОЗУ она попутно запоминается в кэш-памяти, причем «на всякий случай» запоминается информация из адресуемой и соседней с ней ячеек динамического ОЗУ. Так как программы обычно имеют циклический характер, а данные и команды размещены в соседних ячейках, в кэш-памяти постепенно накапливаются, а затем автоматически обновляются актуальные коды команд и данных, которыми процессор недавно пользовался. Иными словами, в кэш-памяти накапливаются рабочие копии слов, хранимых в динамическом ОЗУ, причем считывание этих копий производится примерно в 10 раз быстрее, чем если бы считывались слова-оригиналы. При записи информации выигрыша во времени нет, так как новая информация должна обязательно попасть в динамическое ОЗУ.

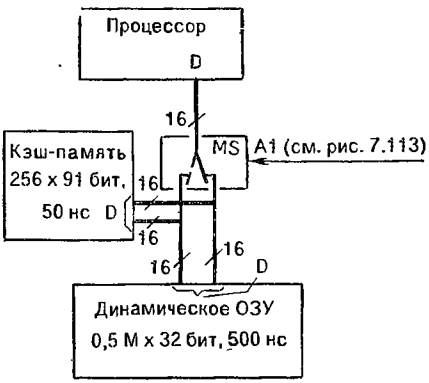


Рис. 7.112. Система с кэш-памятью.

Шины адреса и управления не показаны. Двунаправленный мультиплексор MS управляется разрядом A1 адреса, поступающего из процессора

В табл. 7.3 приведены условия сохранения и обновления информации в ячейках кэш-памяти и динамического ОЗУ [182].

Таблица 7.3

Режим работы	Наличие копии ячейки динамического ОЗУ в кэш-памяти	Информация	
		В ячейке кэш-памяти	В ячейке динамического ОЗУ
Чтение	Копия есть	Не изменяется	Не изменяется
	Копии нет	Обновляется (создается копия)	»
Запись	Копия есть	Обновляется	Обновляется
	Копии нет	Не изменяется	»

Если процессор намерен получить информацию из некоторой ячейки динамического ОЗУ, а копия содержимого этой ячейки уже имеется в кэш-памяти (первая строка таблицы), то вместо оригинала считывается копия; время считывания составляет 50 нс. Информация в кэш-памяти и динамическом ОЗУ не изменяется. Если копии нет, то производится обращение к динамическому ОЗУ с временем считывания 500 нс. Полученная информация пересылается в процессор и попутно запоминается в кэш-памяти вместе с информацией из соседней ячейки (из динамического ОЗУ одновременно считывается пара 16-разрядных слов). Чтение информации в отсутствие копии отражено во второй строке таблицы. Информация в динамическом ОЗУ не изменяется.

При записи информации в ячейку динамического ОЗУ и наличии копии этой ячейки в кэш-памяти (третья строка таблицы) оригинал и копия обновляются. Соседние 16-разрядные ячейки как в кэш-, так и в динамической памяти не затрагиваются и хранят старую информацию. Если копии нет (последняя строка таблицы), то обновляется только содержимое ячейки динамического ОЗУ, т. е. копия не создается.

Таким образом, кэш-память ускоряет считывание команд и данных, которые «недавно» считывались или корректировались (первая строка табл. 7.3), в остальных ситуациях (вторая — четвертая строки таблицы) выигрыша нет. Однако и этого достаточно для повышения производительности системы. Согласно [182], при выполнении программ процессором операции записи в память составляют 10 %, а операции чтения — 90 % общего числа обращений к памяти. В свою очередь, при использовании кэш-памяти (рис. 7.113) от 80 до 95 % операций чтения выполняются быстро (за 50, а не 500 нс). Поэтому среднее время считывания составляет $0{,}9 \cdot 0{,}8 \cdot 50 + (1 - 0{,}9 \cdot 0{,}8) \cdot 500 \approx 180$ нс.

В примере, приведенном на рис. 7.113, кэш-память содержит два одинаковых блока ОЗУ (первый и второй) и одноразрядную память признаков S «недавнего использования» блоков. В каждом блоке имеются разряды для хранения кода Tag (англ. «ярлык», «этикетка»), который отображает старшую часть адреса ячейки-оригинала, разряды D, в которых содержится копия 32-разрядной ячейки ОЗУ, и разряд V — признак истинности информации в ячейке блока.

Адрес, поступающий из процессора, интерпретируется так. Младший (нулевой) разряд определяет выбор байта при выполнении процессором байтовых операций — если в этом разряде нуль, то выбирается младший байт 16-разрядного слова, если единица, то выбирается старший байт. При работе с 16-разрядными словами в нулевом разряде адреса присутствует 0, т. е. используются только четные адреса. Первый разряд адреса служит для выбора одного из двух 16-разрядных слов в 32-разрядных ячейках динамического ОЗУ и в ячейках кэш-памяти. Если в первом разряде нуль, то выбирается правое, если единица — левое 16-разрядное слово.

Группа из восьми разрядов, со второго по девятый, определяет адрес одной из 256 91-разрядных ячеек кэш-памяти. Старшие разряды, с 10-го по 21-й, при считывании информации из кэш-памяти сравниваются с соответствующими разрядами кода Tag из выбранной ячейки первого и второго блоков. При записи новой информации в первый или второй блок кэш-памяти разряды 10—21 адреса, поступившие из процессора, запоминаются в качестве нового кода Tag, сопровождающего (в качестве «ярлыка») новую информацию.

Отметим, что процессор «не знает» о столь оригинальной трактовке смысла разрядов адреса (за исключением нулевого); с его точки зрения существует обычное адресное пространство объемом $2^{22}=4M$ байт, наполовину заполненное ячейками динамического ОЗУ с адресами 000000—1FFFFF (объем ОЗУ можно было бы увеличить без внесения каких-либо доработок в кэш-память). Иными словами,

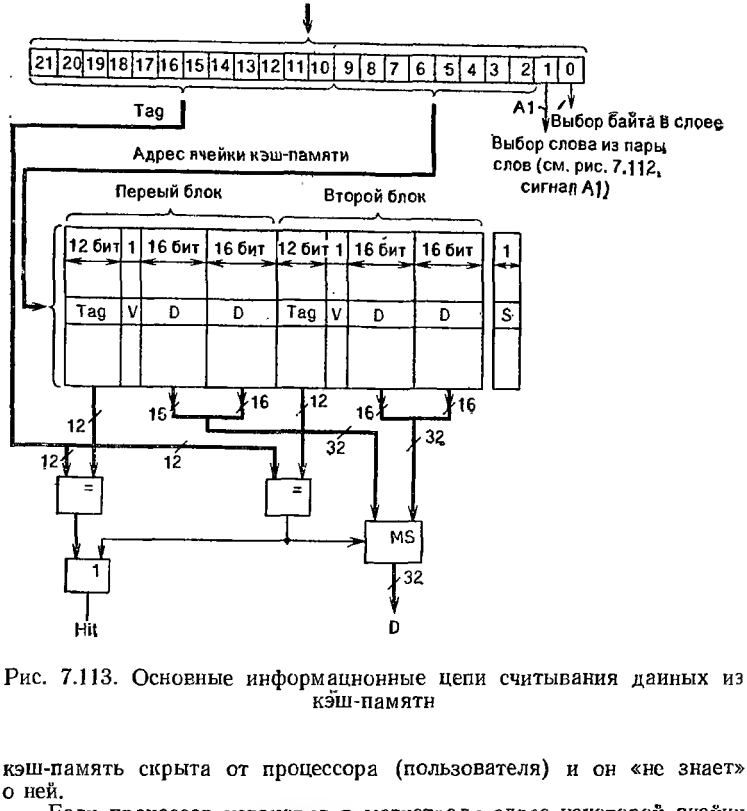


Рис. 7.113. Основные информационные цепи считывания данных из кэш-памяти

кэш-память скрыта от процессора (пользователя) и он «не знает» о ней.

Если процессор установил в магистрали адрес некоторой ячейки динамического ОЗУ и признак чтения, то дальнейшие события развиваются так.

1. Разряд A1 адреса настраивает мультиплексор MS (см. рис. 7.112) на подключение 16-разрядной шины D данных процессора к одной из половин 32-разрядной шины данных ОЗУ — кэш-памяти.

2. Одновременно с этим с помощью разрядов 2—9 адреса выбирается одна из 256 ячеек кэш-памяти. На выходы накопителя кэш-памяти поступает информация из обоих блоков, а именно коды Tag, биты V, а также данные D (по 32 бит из каждого блока).

3. С помощью двух 12-разрядных компараторов сравниваются

старшие разряды поступившего из процессора адреса с соответствующими разрядами, считанными из первого и второго блоков. Если совпадение не зарегистрировано ни первым, ни вторым компаратором, то это означает, что копия затребованной ячейки динамического ОЗУ в кэш-памяти отсутствует. Если произошло совпадение кодов на входах одного из компараторов, то вырабатывается сигнал Hit=1 и через мультиплексор MS в 32-разрядную шину данных посылается пара слов из соответствующего блока кэш-памяти, истинность которой подтверждается соответствующим магистральным сигналом ответа. Отметим, что одновременного совпадения кодов на входах обоих компараторов быть не может — это равносильно тому, что копия создается в двух экземплярах, в действительности этого не происходит и копия является единственной.

4. Если Hit=1, то процессор получает требуемые данные, а разряд S кэш-памяти устанавливается в 0 или 1 в зависимости от того, из какого блока была выдана информация в процессор. Этот разряд, таким образом, отражает очередность обращения к блокам, и по нему можно определить, какая информация более «свежая». На этом операция чтения завершается. Отметим, что нулевой разряд адресного кода при чтении не имеет значения. В процессор посылаются два байта, и если ему нужен только один, то второй им просто не используется. Обращения к динамическому ОЗУ при Hit=1 нет. (Можно было бы «на всякий случай» начать цикл обращения к динамическому ОЗУ, пока не известен результат сравнения, однако прервать этот цикл в произвольный момент нельзя, так как для микросхем динамической памяти существуют ограничения на минимальную длительность сигналов RAS и пауз между ними. Поэтому следует учесть эти ограничения при выборе схемного решения.)

5. Если Hit=0, то разряд S остается без изменения, схема управления (на рис. 7.113 не показана) осуществляет выбор из динамического ОЗУ 32-разрядного слова по адресу, определяемому разрядами 2—21 (напомним, что из динамического ОЗУ одновременно извлекается требуемое 16-разрядное слово и соседнее с ним, правое или левое). Считанное из динамического ОЗУ 32-разрядное слово, согласно строке 2 табл. 7.3, должно быть записано в кэш-память. Адрес ячейки известен — он определяется разрядами 2—9 кода, установленного процессором в адресной шине магистрали. Чтобы определить, в какой блок (первый или второй) следует записать считанное 32-разрядное слово, анализируется разряд S выбранной ячейки кэш-памяти. В зависимости от его состояния (0 или 1) определяется более «устаревшая» информация, которая затем заменяется новой, после чего состояние разряда S инвертируется.

При записи 32-разрядного кода в ячейку первого или второго блока кэш-памяти активизируются оба его накопителя слов D независимо от значения первого разряда адреса, установленного процессором. В разрядах кода Tag ячейки выбранного блока запоминается группа разрядов 10—21 адреса, поступившего из процессора, признак V устанавливается в 1. Правое или левое 16-разрядное слово считанного из динамического ОЗУ 32-разрядного кода (в соответствии со значением первого разряда адреса) посылается в процессор. На этом операция чтения с попутным созданием новой копии завершается.

В исходном состоянии, после включения напряжения питания или после работы канала прямого доступа в память, все признаки V устанавливаются схемой управления в 0, при этом возможные совпадения кодов на входах компараторов считаются недействительными. По мере работы процессора ячейки кэш-памяти заполняются правильной информацией, помеченной признаками V=1. Отметим, что для упрощения аппаратуры разряд S может быть исключен из кэш-памяти; в этом случае выбор уничтожаемого 32-разрядного слова в одном из блоков производится жеребьевкой, с использованием генератора одноразрядного случайного числа с рав-

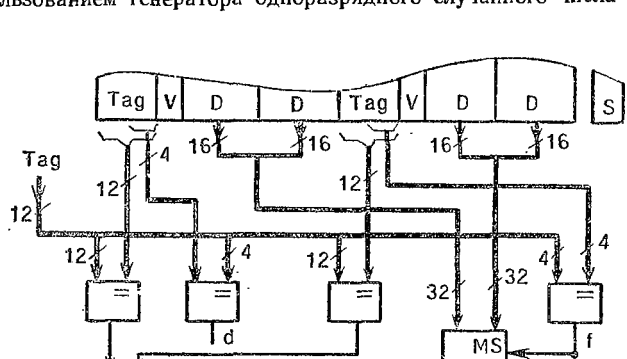


Рис. 7.114. Кэш-память с повышенным быстродействием.

Для управления мультиплексором данных использован не 12-, а 4-разрядный компаратор разрядов адреса

новероятным появлением на его выходе 0 и 1. Однако такой способ приводит к большему числу конфликтов, связанных с уничтожением актуальной информации. Число блоков в кэш-памяти может быть бо́льшим двух для уменьшения вероятности «незаконного» вытеснения новой полезной информацией столь же полезной и актуальной «старой» информации.

Предлагается уменьшить задержку срабатывания кэш-памяти за счет более ранней выдачи данных в процессор, рис. 7.114 [183]. Мультиплексор данных MS управляется сигналом с выхода компаратора, однако компаратор имеет меньшую задержку, чем в схеме рис. 7.113, так как уменьшена разрядность сравниваемых кодов (в данном примере до четырех). Сигнал Hit по-прежнему вырабатывается объединением по схеме ИЛИ сигналов с выходов 12-разрядных компараторов, т. е. задержка формирования этого сигнала осталась прежней. Однако данные поступают в процессор с бо́льшим упреждением, компенсирующим переходные процессы в магистрали и другие факторы, ограничивающие скорость передачи информации.

На первый взгляд, из схемы рис. 7.114 следует, что сравнение кодов для управления мультиплексором данных оказывается неполноценным, так как анализируются не все, а только младшие разряды этих кодов. Действительно, если сигналы d=1 и f=1 сформируются одновременно, то возникает неопределенность выбора направления передачи данных через мультиплексор и любое выбранное направление может оказаться ошибочным. Чтобы таких ситуаций не было, схема управления записью информации в кэш-память следят за тем, чтобы в одноименных ячейках разных блоков кэш-памяти было ходов Tag, совпадающих по четырем младшим разрядам.

Это достигается следующим образом. При создании копии (см. строки 2 и 3 табл. 7.3), на этапе выбора кандидата на уничтожение, блок управления учитывает значения сигналов d и f. Если совпадения нет (d=0, f=0), то принимается во внимание только признак S и уничтожается более «старый» код, а на его место помещается новый, после чего значение S меняется на противоположное (как и в схеме рис. 7.113). Если произошло совпадение кодов на входах одного из компараторов (d=1, f=0 или d=0, f=1), то признак S не учитывается и новый код помещается на место того, который вызвал совпадение (уничтожается «похожий на входящего», остается «отличимый»). При d=1, f=0 новое слово безусловно помещается в первый блок, а при d=0, f=1 — во второй. Таким образом, достигается правильная работа системы, несмотря на неполноценное сравнение кодов. Отметим, что быстродействующие 4-разрядные компараторы могут являться составными частями 12-разрядных.

7.46. Магистраль с электрическими характеристиками, не зависящими от числа подключенных к ней устройств [185]

Чтобы обеспечить такую независимость (рис. 7.115), предлагается передатчики и приемники конструктивно разместить на основной плате в непосредственной близости от линий основной магистрали (на расстоянии не более 6 мм от линии), а платы функциональных устройств подключать через разъемы, установленные на основной плате. Линии основной магистрали согласованы на физических концах резисторами.

Такое решение позволяет развязывать основную магистраль с ответвлениями M_1 от нее. Электрические характеристики магистрали не ухудшаются по мере заполнения разъемов при наращивании системы, как это бывает в «обычных» микроЭВМ, так как в данном случае число передатчиков и приемников, подключенных к основной магистрали, не меняется при изменении конфигурации системы.

В ответвлениях M_1 используются однонаправленные линии связи, согласованные со стороны приемников. В платах функциональных устройств нет необходимости размещать передатчики и приемники вблизи разъема, как это приходится делать для уменьшения неоднородности, вносимой в «обычную» магистраль (см. гл. 4).

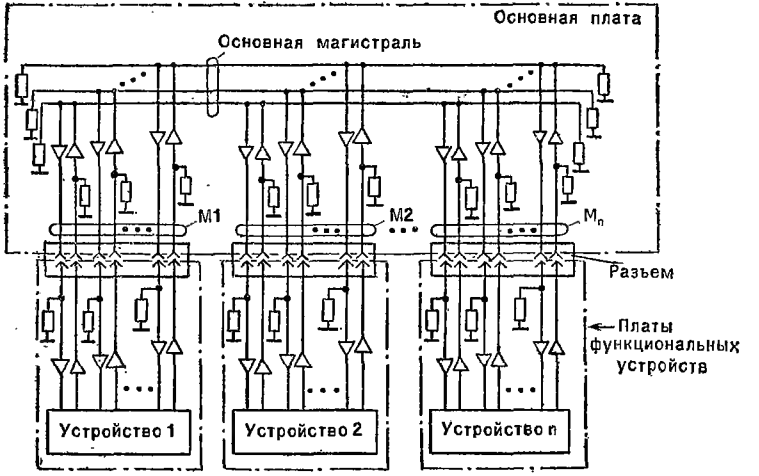


Рис. 7.115. Подключение функциональных устройств к основной магистрали.

Электрические параметры основной магистрали не зависят от числа плат функциональных устройств, установленных в разъемы (число обслуживаемых магистралью разъемов остается постоянным, а заняты они или не заняты — не имеет значения)

7.47. Арбитр двухпортового ОЗУ [188]

В схеме рис. 7.116, а доступ к блокам 0—7 ОЗУ производится через порты 1 и 2 от двух независимых активных устройств. При обращении к одному и тому же блоку возникают конфликты, которые устраняются арбитром. Три старших разряда адреса A21—A19, определяющие номер блока, рис. 7.116, *б*, поступают в арбитр с двух направлений и, если их значения совпадают, то арбитр посылает одному из активных устройств сигнал Блок занят. Этот сигнал поступает в «опоздавшее» к моменту арбитража активное устройство, а при строго одновременных обращениях — в устройство, выбранное случайным образом. Если номера запрашиваемых блоков разные, то доступ к ним производится одновременно через оба порта, так как в этом случае конфликтов не возникает.

Арбитр (рис. 7.117) содержит шесть элементов задержки, каждый из которых может быть выполнен в виде двух последовательно включенных инверторов, шесть элементов Исключающее ИЛИ (М2) и два элемента ИЛИ — НЕ, соединенных по схеме триггера. В отсутствие запросов доступа к ОЗУ со стороны обоих активных устройств сигналы C=1 и D=1 вызывают формирование сигналов A=0 и B=0, что соответствует отсутствию запрета доступа к ОЗУ со стороны обоих активных устройств.

Предложим, что по нижнему каналу поступил некоторый ад-

ресный код, а верхний канал по-прежнему пассивен. Истинность адресного кода подтверждается сигналом лог. 0 на линии запроса (C=0), несколько задержанным относительно адресных сигналов; сигнал D остается равным 1. Поэтому B=0, в точке A формируется нулевой или единичный (при совпадении «полезного» кода A'21 A'20 A'19 с «пассивным» кодом A21 A20 A19) сигнал в зависимости от результатов суммирования сигналов тремя верхними элементами M2. Верхнее активное устройство «не интересуется»

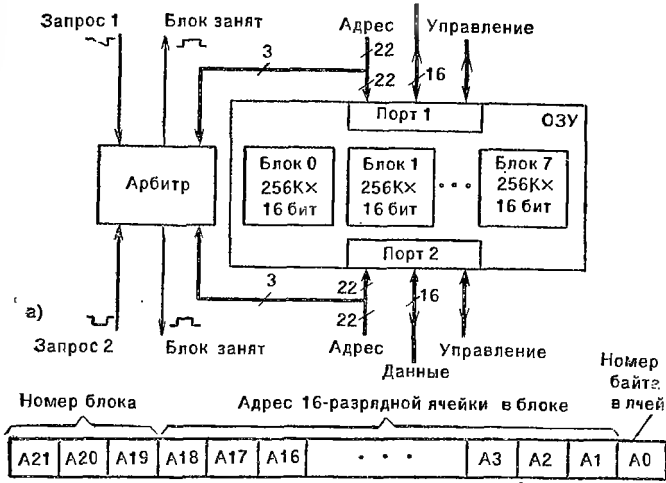


Рис. 7.116. Двухпортовое ОЗУ *(а)* и распределение разрядов адреса *(б)*

сигналом в точке A, так как оно не сформировало сигнал запроса доступа к ОЗУ. Нижнее активное устройство после некоторой задержки, достаточной для окончания переходных процессов в арбитре (в конфликтных ситуациях, описанных далее), проверяет возможность доступа к ОЗУ. В данном случае B=0, следовательно, запрета нет и устройство начинает работу с выбранным блоком ОЗУ.

Предположим далее, что пока нижнее активное устройство работает с ОЗУ, по верхнему каналу поступили адрес и с некоторой задержкой сигнал запроса. Если обращение по верхнему каналу производится не в тот же блок, что и по нижнему, то к моменту поступления сигнала D=0 по крайней мере на одном из трех нижних входов нижнего элемента ИЛИ — НЕ присутствует сигнал несовпадения (лог. 1), следовательно, сигнал B остается равным 0. То же относится и к верхнему элементу ИЛИ — НЕ — в установившемся режиме, после окончания переходных процессов на выходах трех верхних элементов M2, по крайней мере на одном из трех верхних входов элемента ИЛИ — НЕ, устанавливается сигнал лог. 1, а в точке A — лог. 0. Верхнее активное устройство через некото-

рый интервал времени после подачи запросного сигнала (D=0) опрашивает сигнал в точке А, убеждается в том, что блок не занят (А=0) и приступает к работе с ним.

Предположим теперь, что пока нижнее активное устройство работает с некоторым блоком ОЗУ, по верхнему каналу поступил адрес того же блока. Так как адрес с нижнего канала был установлен заранее (вариант одновременного поступления адресов будет рассмотрен далее), сначала срабатывает группа из трех верхних, а затем из трех нижних элементов М2; на выходах этих эле-

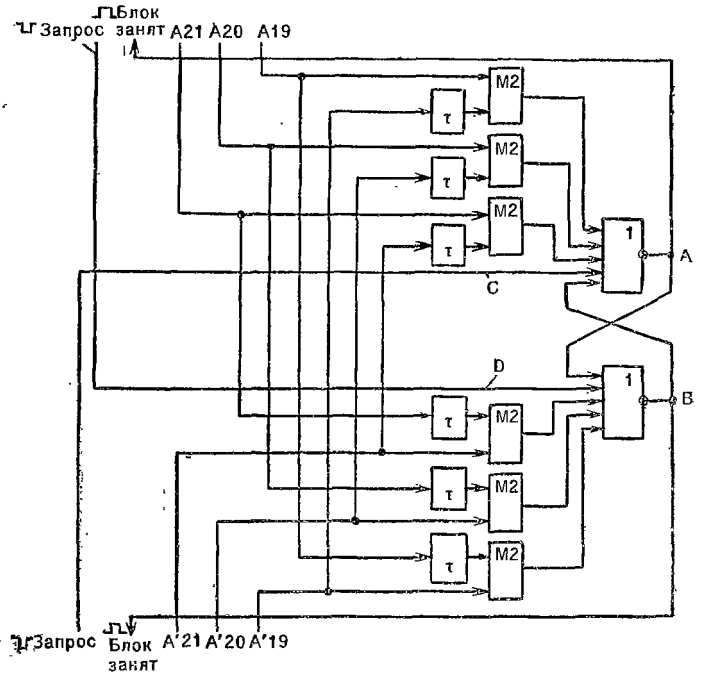


Рис. 7.117. Функциональная схема арбитра (см. рис. 7.116)

ментов появляются сигналы совпадения (лог. 0). Поэтому формируется сигнал А=1, который проходит на вход нижнего элемента ИЛИ — НЕ и предотвращает его ложное срабатывание при переходе сигнала D из 1 в 0. Таким образом, элементы задержки препятствуют возникновению «гонок» при последовательном конфликтном обращении активных устройств к одному и тому же блоку. «Опоздавшее» (в данном случае — верхнее) активное устройство получает сигнал Блок занят, который поддерживается до тех пор, пока конкурирующее активное устройство не закончит обмен.

При строго одновременном поступлении сигналов обращения к одному и тому же блоку со стороны разных активных устройств

на всех входах обоих элементов ИЛИ — НЕ одновременно формируются сигналы лог. 0. Поэтому триггер «предоставляется самому себе» и, благодаря положительной обратной связи, после возможных кратковременных переходных процессов устанавливается в устойчивое состояние A=0, B=1 или A=1, B=0. (Длительность возможных переходных процессов уменьшается с увеличением асимметрии задержек элементов ИЛИ — НЕ.) Таким образом, одно из устройств к моменту опроса арбитра получает сигнал Блок занят и переходит в состояние ожидания.

Арбитр может работать и в отсутствие запросных сигналов (если соответствующие линии исключить из схемы, а освободившиеся входы элеметов ИЛИ — НЕ подключить к шине лог. 0), однако при этом не должно возникать ложных сравнений с кодом, соответствующим пассивному состоянию адресных линий. Например, код 1 1 1 рассматривается как нерабочий, а число блоков ОЗУ сокращается от 8 до 7.

Арбитр можно усовершенствовать, чтобы он обслуживал не два, а три (или более) канала. При этом производится попарное сравнение обращений от разных каналов, результаты сравнения обрабатываются выходными логическими элементами.

7.48. Конструкция магистрали, допускающая ее разделение на изолированные друг от друга части [193]

Общая магистраль, объединяющая функционально-законченные одноплатные устройства, обычно выполняется на основе печатной платы с установленными на ней разъемами для подключения этих устройств. Одноименные контакты всех разъемов объединяются линиями общей магистрали. Число разъемов постоянно и выбирается из расчета на максимальную конфигурацию системы.

На практике, однако, максимальные конфигурации используются далеко не всегда. Обычно занята лишь часть разъемов, остальные свободны. В то же время часто необходимо построить вторую, третью систему устройств, каждая из которых содержит свою магистраль. Поэтому возникает задача использования свободных разъемов для построения второй, третьей и т. д. систем, изолированных по сигнальным цепям друг от друга и от основной

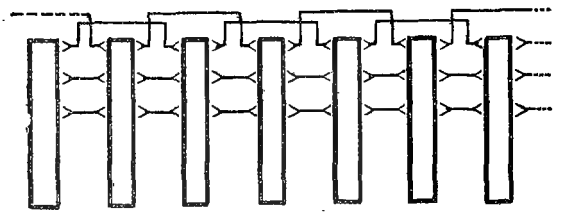


Рис. 7.118. Конструкция линии общей магистрали с возможностью ее разделения.

Прямоугольниками обозначены разъемы, в которые устанавливаются платы

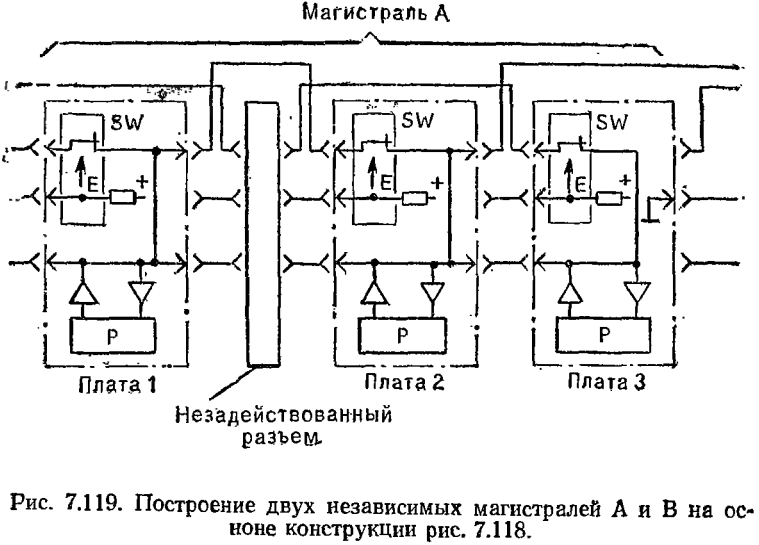


Рис. 7.119. Построение двух независимых магистралей А и В на основе конструкции рис. 7.118.

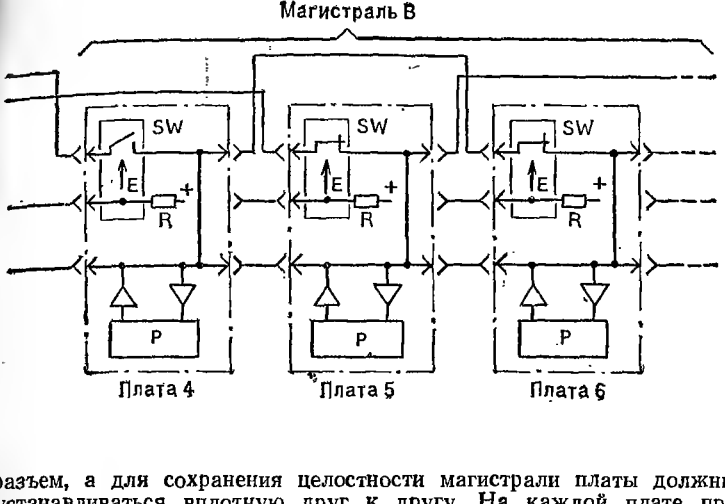
Р — устройства, между которыми осуществляется обмен информацией по магистрали А или В

системы. Иными словами, желательна возможность деления магистрали на произвольные изолированные части.

Такая возможность осуществляется при использовании магистрали рис. 7.118. Пример ее деления на две независимые магистрали А и В приведен на рис. 7.119. В платах установлены электронные или механические (типа реле) ключи SW, управляемые сигналами в точках Е. При Е=0 ключ SW разомкнут, при Е=1 замкнут. Магистраль А включает платы 1—3 и свободный разъем. Плата 3 — концевая, в ее правой группе контактов задействован только средний контакт. Он соединен с шиной лог. 0, так что ключ SW в плате 4 постоянно разомкнут. В результате магистраль В оказывается изолированной от магистрали А.

Незадействованный разъем магистрали А не создает разрыва сигнальной связи благодаря обходной цепи, соединяющей платы 1 и 2. Если необходимо удалить концевую плату 3, то сначала нужно удалить плату 2, чтобы не произошло замыкания магистралей. Можно не использовать концевые платы, если между магистралями оставлять не менее двух свободных разъемов. При этом ключи SW во всех платах должны быть постоянно замкнуты, т. е. они заменяются перемычками на платах; средние контакты плат освобождаются, вместо двух оставшихся правых (или левых) контактов используется один (прямая и обходная связи соединяются между собой с правой или левой стороны разъема), всего на плате остаются три сигнальных контакта.

Если исключить обходные связи и ключи SW, то между магистралями должен оставаться по меньшей мере один свободный



разъем, а для сохранения целостности магистрали платы должны устанавливаться вплотную друг к другу. На каждой плате при этом используются два сигнальных контакта.

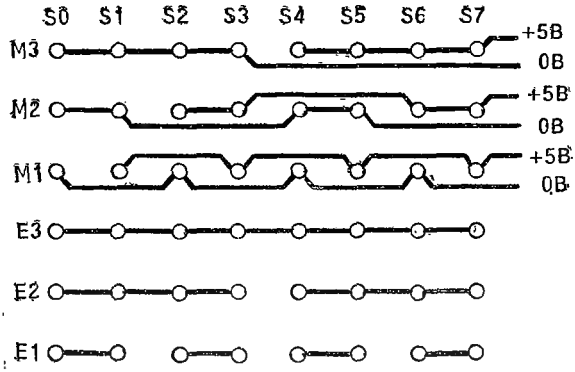
7.49. Быстродействующий распределенный арбитр [194]

На рис. 7.120, *а* условно показан фрагмент топологии печатной платы с разъемами S0—S7 (вертикальными группами из шести контактов). В эти разъемы устанавливаются одноплатные устройства, способные бороться за право доступа к общему ресурсу, например к общей магистрали. Горизонтальные ряды М3—М1 контактов разъемов S0—S7 служат для задания номера одноплатного устройства (код М3 М2 М1). Устройству, установленному в разъем S0, присваивается код М3 М2 М1 = 0 0 0, следующему — 0 0 1 и т. д. Ряды контактов Е3—Е1 и соответствующие связи между ними служат для оперативного обмена информацией между платами при устранении конфликтов. Цепи синхронизации не показаны.

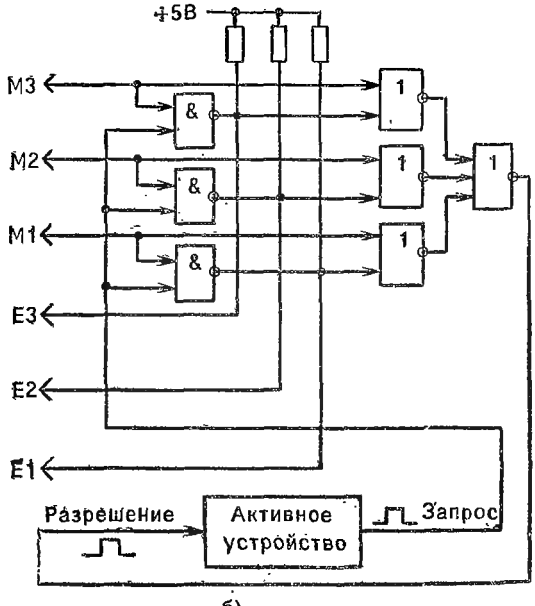
В каждой плате размещена схема (рис. 7.120, *б*), которая при наличии сигнала запроса активного устройства выдает в линии Е3, Е2, Е1 обратный код номера платы и анализирует результирующий код в этих линиях с учетом возможного «противодействия» более приоритетных плат — соседей справа. Элементы И—НЕ выполнены по схеме с открытым коллектором. Устройство, получившее сигнал разрешения, приступает к использованию общего ресурса.

В схеме на рис. 7.121 собственный номер платы не используется. Приоритет платы, установленной в разъем S7, максимален, приоритеты остальных плат убывают по мере продвижения справа налево.

Фрагменты топологии, представленные на рис. 7.120, *а* и 7.121, *а*, имеют регулярную структуру и могут наращиваться по мере увеличения числа плат. Возможна, однако, и иная, много-



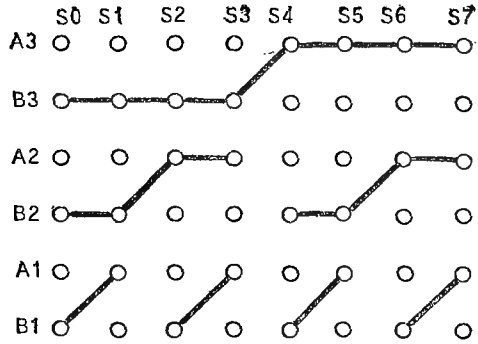
а)



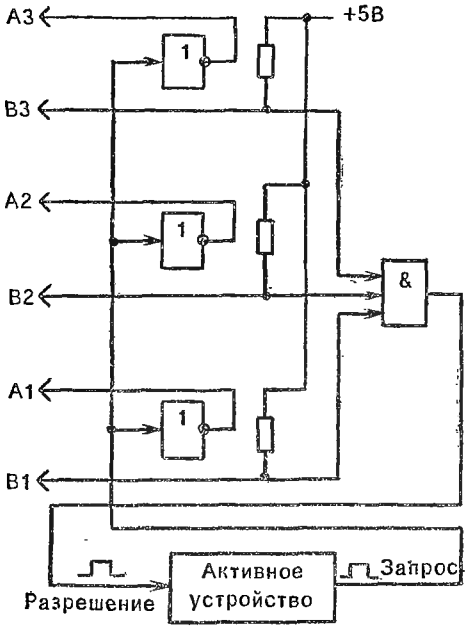
б)

Рис. 7.120. Первый вариант построения системы с приоритетным взаимодействием одноплатных устройств:

а — фрагмент топологии печатной платы, объединяющей одноплатные устройства; б — приоритетная схема, размещенная в каждом одноплатном устройстве



а)



б)

Рис. 7.121. Второй вариант построения системы с приоритетным взаимодействием одноплатных устройств

уровневая структура, в которой устройства разделены на группы, причем арбитраж одновременно ведется как между группами, так и в пределах каждой группы (примерно так же, как распространяется сигнал переноса в сумматорах с групповым переносом).

7.50. Ускорение программно-управляемой пересылки данных при использовании операции фиктивного чтения [204]

Программно-управляемая пересылка данных между памятью микроЭВМ и адресуемым регистром RG контроллера (рис. 7.122) обычно выполняется так. В исходном состоянии процессор работает по фоновой программе, не связанной с обслуживанием контроллера. Контроллер выполняет заданную ему ранее команду считывания информации с гибкого магнитного диска (описанная далее идея использования фиктивного чтения в равной мере применима к записи информации на диск, а в общем случае — к пересылкам информации между любыми адресуемыми элементами микроЭВМ).

Рис. 7.122. Схема микроЭВМ.

Однотактная пересылка информационного слова из регистра RG в ОЗУ производится минуя процессор, но при его участии («без помощи, но при посредстве», см. штриховую стрелку)

Подготовив в регистре RG очередное информационное слово D, считанное с диска, контроллер формирует сигнал готовности, который поступает на вход прерывания процессора. Процессор временно откладывает выполнение фонового задания, запоминает в ОЗУ параметры возврата к нему и приступает к работе по прерывающей программе, которая предусматривает выполнение следующих действий.

1. Считывание слова данных D из регистра RG контроллера в один из регистров общего назначения процессора (см. стрелку 1 на рис. 7.122).

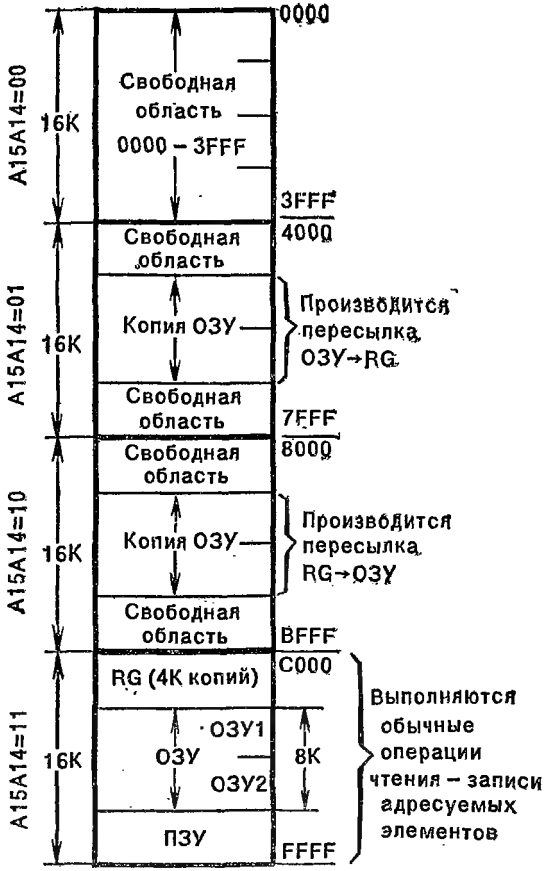


Рис. 7.123. Распределение адресного пространства микроЭВМ.

Копии ОЗУ в областях 01 и 10 используются только для передачи данных между ячейкой ОЗУ и регистром RG (см. рис. 7.122)

2. Считывание из ОЗУ указателя адреса, по которому следует переслать слово данных D.

3. Запись слова данных D в ячейку ОЗУ в соответствии с текущим значением указателя адреса (см. стрелку 2).

4. Прибавление константы (равной 2 при байтовой адресации и разрядности кода D, равной 16 бит) к указателю адреса и запись полученного числа в ячейку ОЗУ для использования в следующем цикле обслуживания контроллера.

После выполнения этих действий процессор возвращается к фоновому заданию. Если поток информационных слов, поступаю-

щих из регистра RG контроллера, не слишком интенсивен, то процессор успевает пересылать эти слова в ОЗУ, в противном случае была бы возможна потеря информации.

Чтобы ускорить выполнение операций пересылки, предлагается вместо двух передач, отмеченных на рис. 7.122 стрелками 1 и 2, делать одну, условно показанную на рисунке штриховой стрелкой. Передача информационного слова D производится минуя процессор, но при его участии. Для пояснения этой идеи рассмотрим микроЭВМ более подробно.

Предположим, что адресное пространство микроЭВМ имеет объем 64К, объем ПЗУ—4К, ОЗУ—8К ячеек. Как следует из рис. 7.123, все адресное пространство делится двумя старшими разрядами A15 и A14 адреса на четыре области по 16К ячеек: 00, 01, 10 и 11. Область 00 используется для адресации элементов, не показанных на рис. 7.122 и не имеющих отношения к существу предлагаемого решения («свободная область»). Область 11 рассматривается как основная, в которой размещены адреса ПЗУ, ОЗУ и 4К копий адреса регистра RG. Эта область, в отличие от оставшихся, не имеет каких-либо особенностей и используется для обычной адресации размещенных в ней элементов

В областях 01 и 10 размещены копии ОЗУ, однако эти копии в некотором смысле далеко не равноценны своему оригиналу из области 11, как это было бы при использовании обычного метода частичной дешифрации адресов, см. § 1.3. Точнее, области 01 и 10 используются для реализации ускоренных пересылок соответственно из ячейки памяти в регистр RG и из регистра RG в ячейку памяти.

Благодаря рассмотренному далее нестандартному схемному решению блока дешифрации и управления, микроЭВМ приобрела новые и полезные свойства. При выполнении процессором операции чтения информации из ячейки памяти с адресом, принадлежащим области 01, считанное из ОЗУ слово передается по магистрали непосредственно в регистр RG контроллера. Это же слово попутно попадает и в процессор, но не используется им, так как полезная пересылка (ОЗУ — регистр RG) уже произошла.

Аналогично при выполнении процессором операции чтения информации из ячейки памяти, принадлежащей области 10, в действительности производится запись информации, выданной в магистраль из регистра RG, в эту ячейку. Таким образом, при считывании процессором ячеек ОЗУ, размещенных в областях 01 и 10, схемы дешифрации и управления формируют сигналы считывания ОЗУ — записи в регистр RG и считывания регистра RG — записи ОЗУ.

Дешифратор адреса (рис. 7.124) анализирует четыре старших разряда A15—A12 адреса, выдаваемого процессором в магистраль. Память выполнена на микросхеме ПЗУ со структурой 4К×16 бит и двух микросхемах ОЗУ с такой же структурой. Выбор микросхем ПЗУ, ОЗУ и регистра RG контроллера производится при формировании дешифратором адреса единичного сигнала на входе E соответствующей микросхемы. Из схемы рис. 7.124 следует, что регистр RG выбирается как при обычном обращении (при A15=A14=1 и A13=A12=0, см. рис. 7.123), так и при обращении к ОЗУ через области 01 и 10. Выбор ПЗУ возможен только при A15=A14=A13=A12=1. Выбор ОЗУ производится индивидуально

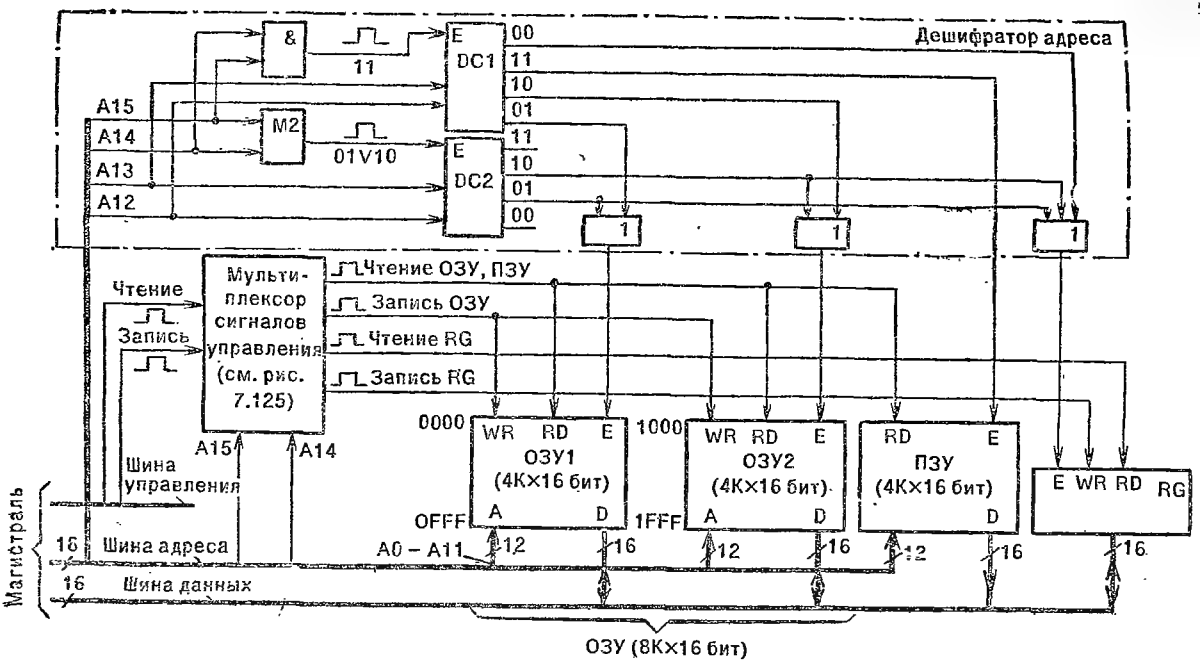


Рис. 7.124. Схема доступа к адресуемым элементам микроЭВМ

при обращении в область 11 или совместно с регистром RG контроллера при обращении к копиям в областях 01 и 10.

Управляющие сигналы чтения — записи адресуемых элементов (регистра RG, ОЗУ и ПЗУ) формируются с учетом, того, какой области принадлежит текущий адрес (рис. 7.125). Если адрес относится к области 11, то мультиплексоры MS транслируют на выходы сигиалы с входов 11, так что сигналы Запись и Чтение проходят из процессора по магистрали и далее через мультиплексоры MS на соответствующие входы управления адресуемых элементов. Если обращение к ОЗУ производится в области 10, то мультиплексоры MS транслируют на выходы сигналы с входов 10. Это приводит к тому, что сигнал чтения, формируемый процессором, поступает на управляющие входы WR записи микросхем ОЗУ и на вход RD чтения регистра RG. Аналогично при чтении информации из ОЗУ в области 01 мультиплексоры MS формируют сигналы считывания для микросхем ОЗУ и ПЗУ (ПЗУ в этой области не выбрано адресным дешифратором, поэтому оно не участвует в обмене) и сигнал записи для микросхемы регистра RG.

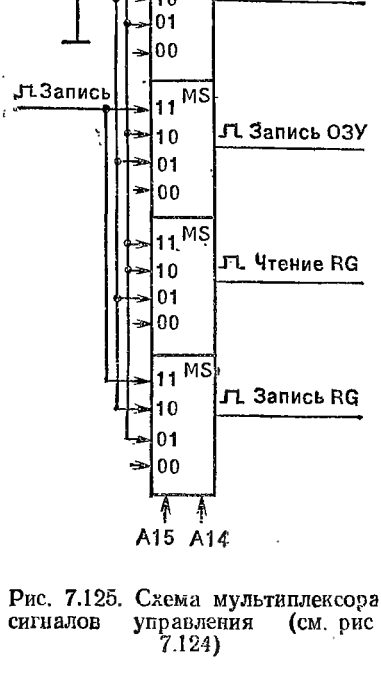


Рис. 7.125. Схема мультиплексора сигналов управления (см. рис 7.124)

В зависимости от применяемой элементной базы, в схему рис. 7.124 могут вводиться дополнительные стробирующие сигналы, элементы задержки и другие средства компенсации возможных «гонок». Схемные решения, подобные приведенному на рис. 7.124, позволяют повышать скорость обмена данными между адресуемыми элементами. Отметим, что если в системе команд процессора имеются двухадресные команды типа Память — память, то и в этом случае предлагаемый механизм непосредственной пересылки работает быстрее, так как для передачи кода нужен только один цикл использования магистрали.

7.51. Подтверждение истинности передаваемых данных попеременно фронтом и срезом синхросигнала [207]

Передаваемый по каналу связи параллельный код данных (адреса) обычно сопровождается синхроимпульсом, подтверждающим его истинность. При этом правильность кода гарантируется либо в течение действия всего синхроимпульса (как в интерфейсе Общая шина, см. рис. 1.6), либо в момент действия его фронта

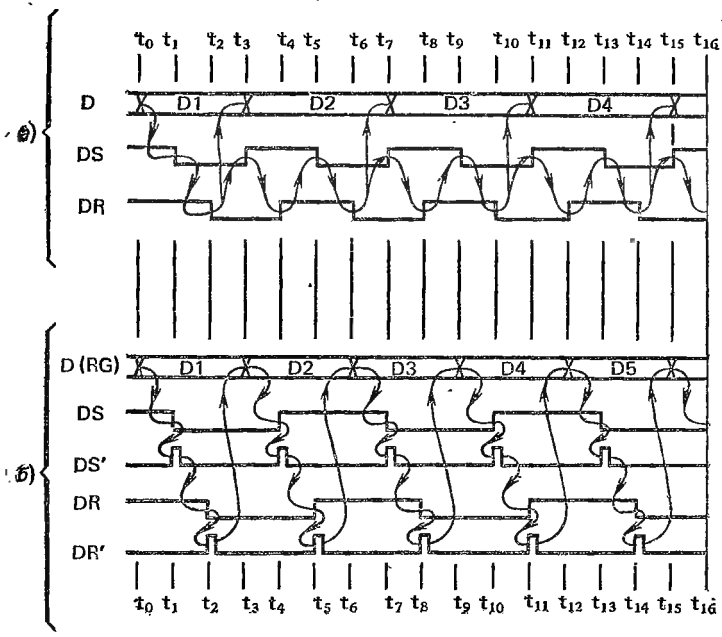


Рис. 7.126. Временные диаграммы передачи данных в известной (*а*) и предлагаемой (*б*) системах связи передатчика с приемником

(рис. 7.126, *а*). Чтобы показать преимущества предлагаемого способа передачи данных [207] (рис. 7.126, *б*), рассмотрим сначала известный способ (см. рис. 7.126, *а*).

В момент t_0 устройство-передатчик формирует в линиях шины данных D канала связи некоторый параллельный код D1. В управляющих линиях DS и DR канала связи присутствуют напряжения высокого уровня (H). После гарантированного установления правильных уровней напряжения во всех линиях шины данных (с некоторым запасом) в момент t_1 передатчик формирует напряжение низкого уровня (L) в линии DS. По фронту сигнала в линии DS

(при переходе сигнала из состояния H в состояние L) устройство-приемник запоминает код D1, поступающий с шины D, и в момент t_2 формирует ответный сигнал в линии DR.

Приняв ответный сигнал, передатчик в момент t_3 готовит очередной код D2 и снимает сигнал с линии DS. Приемник, в свою очередь, в момент t_4 снимает сигнал с линии DR. Далее, в момент t_5 передатчик устанавливает сигнал (L-уровень) в линии DS, подтверждая истинность кода D2, и процесс повторяется. Из рис. 7.126, *а* следует, что за период времени t_0—t_{15} от передатчика к приемнику передано четыре кода D1—D4.

Предлагается увеличить информационную нагрузку линий управления DS и DR более рациональным кодированием сообщений о текущих событиях. В результате более эффективного использования передних и задних фронтов сигналов в линиях DS и DR временна́я диаграмма пересылки кодов от передатчика к приемнику становится более сжатой (см. рис. 7.126, *б*), так что за период времени t_0—t_{15} передаются не четыре, а пять кодов D1—D5. Рассмотрим процесс передачи подробнее, с использованием примера схемного решения, приведенного на рис. 7.127.

Передатчик и приемник выполнены с использованием микроЭВМ1 и микроЭВМ2. Каждая микроЭВМ имеет шину адреса, данных и управления (A, D и C, см. § 2.1). В исходном состоянии программно-доступный по чтению RS-триггер T1 готовности приемника установлен в 1 (готовность имеется), сигналы на его R- и S-входах отсутствуют (равны 0). Программно-доступный по чтению триггер T2 готовности данных, поступивших из передатчика, установлен в 0 (готовности нет), сигналы на его R- и S-входах также отсутствуют. D-триггеры TT1 и TT2, включенные в режиме одноразрядных счетчиков, могут находиться в произвольных состояниях; для определенности предположим, что оба триггера исходно установлены в 1. Формирователи импульсов F вырабатывают на выходах кратковременные положительные импульсы при изменении уровня сигнала на входах (H—L или L—H). Проверив состояние триггера T1, микроЭВМ1 приступает к выполнению пересылки данных в приемник; микроЭВМ2 подготовлена к приему данных и ведет периодический программный опрос триггера T2, дожидаясь готовности первого слова данных. (Возможен обмен с использованием режима прерываний, см. гл. 2, однако для простоты он здесь не рассматривается.)

Программный опрос триггеров T1 и T2 происходит при выполнении микроЭВМ1 и микроЭВМ2 команд пересылки типа Память — регистр. Соответствующие адреса распознаются дешифраторами DC1 и DC2, в результате открываются передающие элементы, подключенные к выходам триггеров, и информация из них поступает в соответствующие микроЭВМ. Более подробно о программно-доступных элементах см. гл. 1, 2.

В момент t_0 (см. рис. 7.126, *б*) микроЭВМ1 записывает данные с шины D своей магистрали в адресуемый регистр RG. Для этого в шину A выдается адрес этого регистра, срабатывает дешифратор DC1 (M=1) и на входе синхронизации регистра RG формируется положительный перепад напряжения, по которому в регистр RG принимается код D1. Сигнал MSYN подтверждает истинность адреса в магистрали (см. § 2.1) и стробирует дешифратор DC1. Сигнал M с выхода дешифратора DC1 устанавливает в 0 триггер T1 и, проходя через элемент ИЛИ — НЕ, формирует ответный сиг-

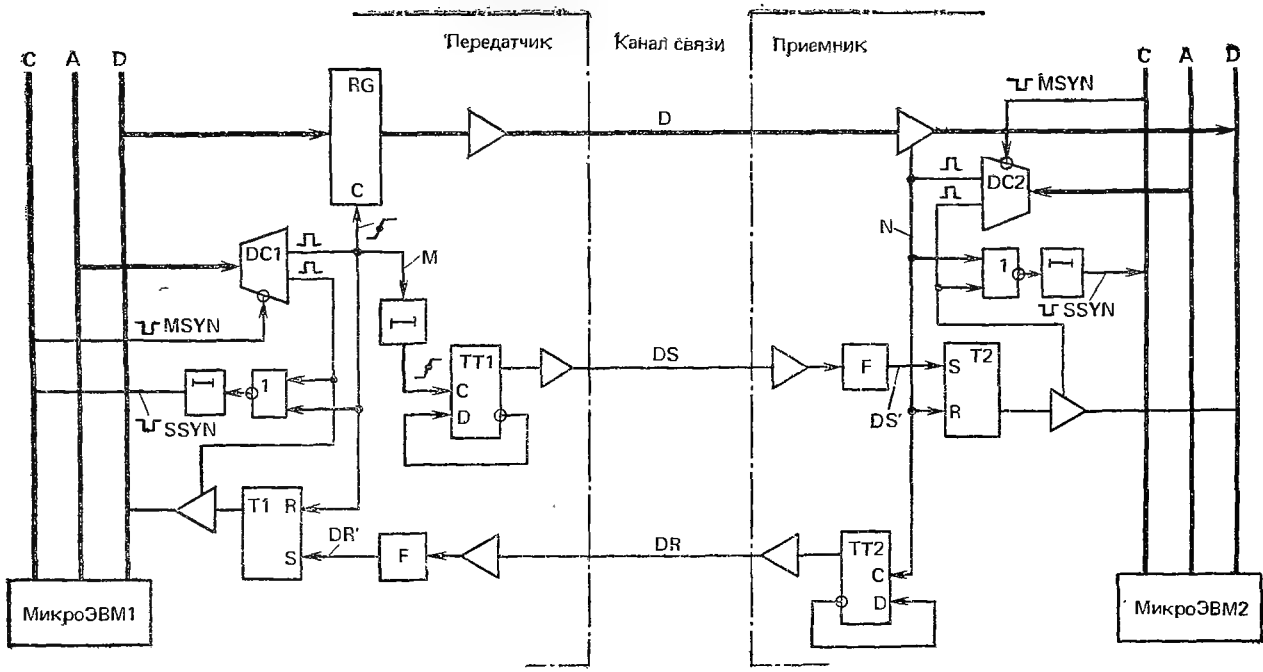


Рис. 7.127. Предлагаемая система связи передатчика с приемником

иал SSYN, подтверждающий выполнение записи в регистр RG. (Линия Чтение — запись шины С может не использоваться для упрощения схемы, так как регистр RG доступен только по записи; это же относится и к другим адресуемым элементам, доступным только по чтению.)

Сигнал М с выхода дешифратора DC1 проходит через элемент задержки и воздействует на вход синхронизации D-триггера TT1, в результате его состояние меняется и в момент t_1 в линии DS формируется напряжение низкого уровня. Формирователь импульсов F приемника по фронту сигнала в линии DS вырабатывает импульс DS′, который устанавливает в 1 триггер Т2.

МикроЭВМ2 при очередном считывании состояния программно-доступного триггера Т2 (с использованием дешифратора DC2, сигналов MSYN, SSYN Общей шины) обнаруживает, что этот триггер установлен в 1. Это означает, что данные поступили на входы приемника. МикроЭВМ2 открывает группу программно-доступных усилителей и считывает через свою шину данных код D1 из регистра RG. При считывании на выходе дешифратора DC2 формируется сигнал N, триггер Т2 устанавливается в 0, триггер TT2 меняет состояние, в линии DR формируется напряжение низкого уровня (момент t_2), на входе S триггера Т1 формируется импульс, этот триггер устаиавливается в 1.

МикроЭВМ1 при очередном опросе триггера Т1 обнаруживает на его выходе сигнал лог. 1 и расценивает этот факт как готовность приемника к получению следующего слова. Поэтому в момент t_3 микроЭВМ1 записывает в регистр RG очередной код (D2). После этого триггер TT1 вновь меняет состояние, и в момент t_4 в линии DS формируется фронт сигнала, на S-входе триггера Т2 формируется импульс, этот триггер устанавливается в 1. После проверки состояния триггера Т2 при считывании кода D2 сигнал N меняет сигнал на выходе триггера TT2 и в момент t_5 в линии DR формируется напряжение высокого уровня. На S-вход триггера Т1 поступает импульс, и далее процессы повторяются.

Из временных диаграмм, показанных на рис. 7.126, *а* и *б*, следует, что в предлагаемой системе передачи информации частота сигналов управления в линиях DS и DR канала связи снижена в 1,5 раза, т. е. снижены требования к частотным характеристикам канала, хотя плотность передаваемой информации более высокая. Сигналы DS′ и DR′ удвоенной частоты не транслируются через канал связи, а используются внутри передатчика и приемника.

Глава 8

Аппаратная поддержка вычислений, цифровые фильтры, узлы микроЭВМ

8.1. Расширение системы команд процессора путем подключения к нему программно-доступных элементов [84]

На рис. 8.1 показана схема подключения дополнительных элементов: дешифратора DC, колодки с перекрестными связями K и элементов И, к шинам процессора с целью однотактной реализации команды перестановки бит. Процессор, «желая» произвести пе-

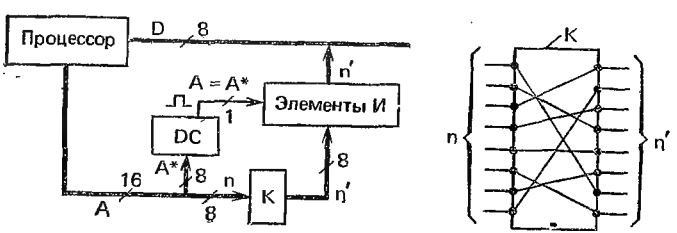


Рис. 8.1. Расширение системы команд процессора. Операнд передается не по шине данных, а по шине адреса

рестановку бит в некотором 8-разрядном коде n, выполняет команду чтения из ячейки памяти с адресом, в старшем байте которого содержится некоторый код A*, а в младшем — код n:

A*	n

Ячейки памяти с таким адресом не существует. На часть адреса A* «откликается» дешифратор, открывая элементы И по первым входам, в результате в шину данных D попадает код n', полученный перестановкой бит в коде n. Затраты адресного пространства для реализации дополнительно введенной команды в данном примере составляют 256 ячеек, по числу возможных комбинаций бит в 8-разрядном коде n.

Вместо операции перестановки бит может использоваться любая другая операция (например, инверсия некоторых разрядов, конъюнкция некоторых разрядов и т. п.), программное выполнение которой требует слишком большого времени.

8.2. Распараллеливание выбора двух операндов при выполнении двухадресных команд [86]

Рассматриваются двухадресные команды типа

КОП	A1	A2

Команда, например, может быть такой: прочитать содержимое ячеек памяти с адресами A1 и A2, сложить полученные числа и записать результат в ячейку памити с адресом A1. Как видим, нужны три обращения в память: два — по чтению и одно — по записи. Предлагается уменьшить число обращений до двух с использованием дублированной памяти (рис. 8.2). Для извлечения первого и второго операндов процессор одновременно запускает ОЗУ1

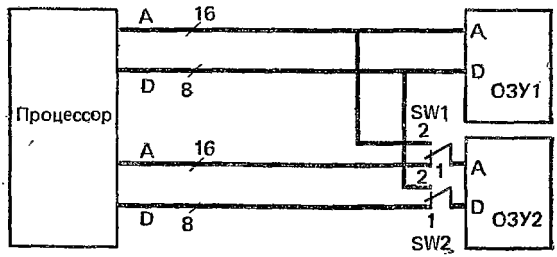


Рис. 8.2. Память с раздельным считыванием и параллельной записью

и ОЗУ2, информация в которых одинакова. Ячейка с адресом A1 выбирается из ОЗУ1, а ячейка с адресом A2—из ОЗУ2. Ключи SW1 и SW2 находятся в положении 1. Под «ключами» подразумеваются группы логических элементов, управляемые от микропрограммного устройства управления процессора. Вычислив результат (в нашем примере — путем сложения полученных чисел), процессор переводит ключи в положение 2 (ОЗУ1 и ОЗУ2 включаются при этом параллельно), после чего выдает адрес A1 и сумму в верхнюю магистраль. Этот результат, таким образом, запоминается по адресу A1 как в ОЗУ1, так и в ОЗУ2.

8.3. Аппаратная поддержка механизма вызова подпрограмм [87]

Если процессор встречает в программе команду вызова подпрограммы, то он временно откладывает выполнение основной программы, запоминает в стеке параметры возврата и переходит к выполнению подпрограммы. Дойдя до конца подпрограммы, процессор встречает команду возврата, которая заставляет его вер-

нуться к основной программе. Обычно для вызова подпрограммы используется команда вида

КОП	A

(16 бит).

Код операции КОП содержит, например, 4-битовую комбинацию разрядов типа 0110, которая расценивается устройством управления процессора как команда «вызов подпрограммы». Последующие разряды команды (поле А) определяют адрес начала подпрограммы, которой должно быть передано управление. Если, например, поле А представлено 12 битами, то команда содержит 16 бит и для ее извлечения из памяти при 8-разрядной шине данных требуются два обращения к памяти. Предлагается «сжать» эту команду в один байт, что позволит извлечь ее из памяти за один цикл. Для этого вместо указания адреса начала подпрограммы в команде указывают 4-битовый номер подпрограммы:

КОП	N

(8 бит).

Такая команда способна отослать процессор к одной из 16 подпрограмм. Адреса начал этих подпрограмм хранятся в специально введенном для этой цели ПЗУ или сверхоперативном запоминающем устройстве (СОЗУ), выполненном, например, на регистрах (рис. 8.3). По номеру подпрограммы из этого ПЗУ или СОЗУ извлекается 12-разрядный адрес начала подпрограммы, который используется для передачи управления.

Использование подобного принципа сжатия информации позволяет уменьшить емкость памяти видеоОЗУ цветного растрового графического дисплея. Если, например, по каждому из основных цветов (красный, зеленый, синий) можно задать $2^8=256$ градаций яркости, то для описания каждой точки экрана нужно было бы иметь 24-разрядную ячейку — по одному байту на каждый цвет.

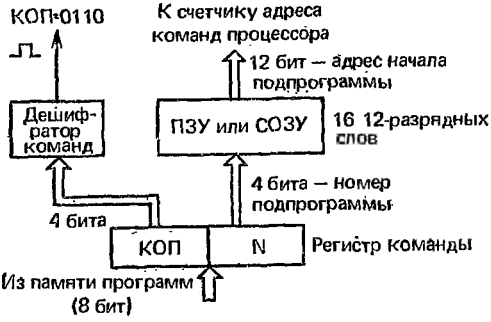
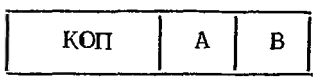


Рис. 8.3. Схема формирования адреса начала подпрограммы

Однако это число можно сократить, если, как это делают художники, ограничиться конечной палитрой красок, содержащей, например, 64 цвета — чистые основные цвета и (или) любые их смеси. В этом случае для описания точки на экране нужно иметь всего 6 разрядов, которые указывают, каким из 64 цветов нужно «раскрасить» эту точку. Свобода выбора палитры при этом огромна — программист вправе выбрать любые 64 цвета из 2^{24}. В данном случае палитра представляет собой СОЗУ, в котором хранятся 64 24-разрядных слова (по одному байту на каждый основной цвет).

8.4. Аппаратная реализация вызова произвольных участков программ, не являющихся «оформленными» подпрограммами [88]

Предлагается ввести в систему команд процессора команду типа



Поле кода операции КОП содержит некоторый код, однозначно определяющий эту команду; поле А содержит адрес начала участка программы, на который должно быть передано управление, поле В — адрес последней команды этого участка, после выполнения которой управление вновь передается основной программе.

Встретив в основной программе такую команду, процессор запоминает адрес последующей команды в стеке 1 (рис. 8.4, стрелка-

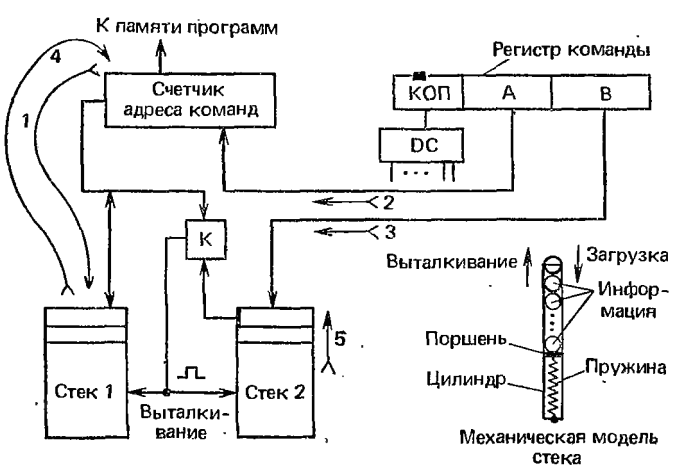


Рис. 8.4. Схема устройства и этапы его работы (указатели 1—5)

указатель 1), а затем передает адрес А начала нового участка программы в счетчик адреса команд (указатель 2) и одновременно с этим адрес конца участка — в стек 2 (указатель 3). После этого процессор обычным путем продвигается по новому участку программы. Выход на конец участка сопровождается срабатыванием компаратора К, в результате из верхушек стеков «выталкивается» запасенная ранее информация. При этом в счетчик адреса команд из стека 1 попадает адрес очередной команды основной программы (сохраненный в стеке 1 ранее, на этапе 1), так что процессор вновь возвращается к основной программе (см. указатель 4). Адрес В при выталкивании из стека 2 теряется (указатель 5). Использование стеков 1 и 2 позволяет работать с вложенными участками программ, т. е. в тех случаях, когда, например, программа П1 вызывает участок программы П10, а тот, в свою очередь, участок программы П13 и т. д. Число таких уровней вложения определяется глубиной стеков 1 и 2.

Предлагаемая команда позволяет повышать гибкость программирования путем использования в качестве подпрограмм участков «чужих» программ, не оформленных в виде подпрограмм, т. е. не заканчивающихся командой Выход из подпрограммы. Команда также упрощает процедуры отладки и диагностики.

8.5. Микропрограммная поддержка механизма выбора из памяти «разбросанных» данных [89]

В ряде задач извлекаемые из памяти данные расположены в ней не в соседних ячейках, а с некоторым шагом. Например, в памяти могут храниться результаты переписи населения, где для каждого человека отводятся 120 ячеек (дата рождения, состав семьи и т. д.). Для выявления, например, долгожителей процессор, помимо выполнения прочих операций, должен сканировать память с шагом 120 ячеек, выбирая каждый раз параметр «возраст». Обычный процессор при этом должен каждый раз прибавлять число 120 к содержимому некоторого регистра, выполняя команду сложения.

Предлагаемый вариант построения процессора не требует выполнения такой команды. Подготовка очередного адреса производится автоматически при отработке микропрограммы, соответствующей новой команде, введенной в систему команд процессора. Структурная схема процессора приведена на рис. 8.5. Смысл данного решения состоит в том, чтобы каждый раз при выполнении указанной команды «подтягивать» на 120 ячеек содержимое индексного регистра, готовясь к выбору из памяти следующего слова. Команда, как обычно, поступает из памяти (на рисунке не показана) в регистр команд. Код операции поступает в устройство управления УУ, которое вызывает следующую последовательность микрокоманд при обращении за операндом:

1. Код «смещения» (в данном случае число 120) поступает из регистра команд в регистр В.

2. Содержимое индексного регистра подается через магистраль на второй вход арифметического устройства АУ, результат сложения (физический адрес операнда) запоминается в регистре А.

3. Содержимое регистра А передается в регистр адреса памяти

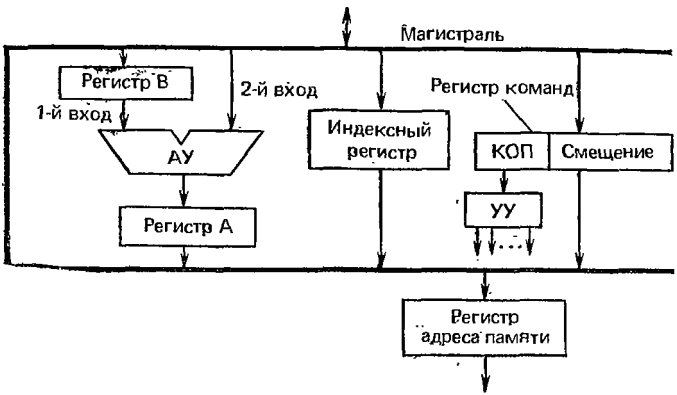


Рис. 8.5. Структурная схема процессора

и одновременно с этим по магистрали — в индексный регистр. Это предопределяет «скачок» на 120 ячеек при последующем получении подобной команды.

8.6. Цифровой фильтр для сглаживания ступенчатых воздействий на механические исполнительные органы [97, 189]

При выдаче управляющих воздействий от микроЭВМ на механические исполнительные органы необходимо обеспечить плавный разгон и торможение движущихся частей, чтобы исключить ударное (скачкообразное) перемещение деталей (например, пера графопостроителя), что вызывает чрезмерные механические и электрические перегрузки, потерю качества отработки воздействий из-за вибрации, инерционности и т. д. Решение задачи сглаживания сигналов аналоговыми средствами не обеспечивает высокой степени интеграции и высокой точности дозировки воздействий.

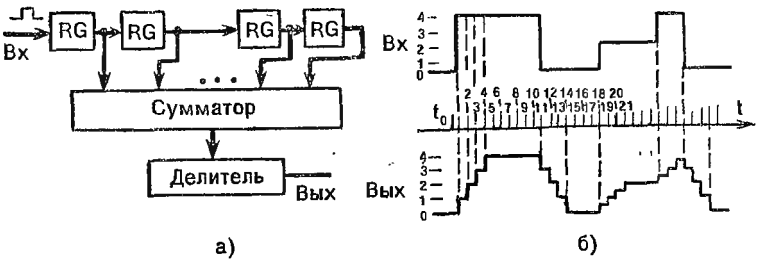


Рис. 8.6. Схема сглаживания ступенчатых сигналов

Рассмотрим решение этой задачи с помощью цифровой схемы (рис. 8.6). Схема содержит последовательно включенные регистры, сумматор и делитель. Число входных регистров, уменьшенное на единицу, определяет число промежуточных градаций при сглаживании ступенчатого сигнала. Сумматор выполняет арифметическое сложение чисел, поступающих на его входы. Делитель усредняет полученную сумму, т. е. делит сумму на число входных каналов сумматора. Если это число равно 2^n, то деление сводится к переносу запятой в полученной сумме на n разрядов влево.

В исходном состоянии (рис. 8.6, *б*, момент t_0) во всех регистрах записаны нули, на выходе сумматора и делителя — нулевые коды. В момент t_1 в первый регистр записывается число 4, а в остальных сохраняются нулевые коды. На выход сумматора выдается число 4, на выход устройства — число 1 (производится деление на 4). В следующий момент t_2 число 4 присутствует как в первом, так и во втором регистрах. Среднее значение суммы равно двум. В момент t_3 на выходе сумматора формируется число 12, а на выходе устройства — число 3. В момент t_4 число 4 заполняет все регистры и среднее значение достигает четырех.

Обратный процесс развивается аналогично. В момент t_{18} на вход поступает число 2. Процесс повторяется, однако шаг квантования по амплитуде автоматически уменьшается вдвое — до 0,5. Вместо набора сдвиговых регистров можно использовать обычные регистры, включенные по входам параллельно. В этом случае входные данные поочередно циклически загружаются в эти регистры.

8.7. Цифровой фильтр для усреднения значений входных сигналов [98]

Устройство, показанное на рис. 8.7, предназначено для определения граничных или промежуточных значений входных сигналов, в частности для их усреднения. В данном примере устройство имеет 7 входов, по которым в параллельном коде поступают 7 близких по значению или равных чисел, например результаты измерения одной и той же величины с помощью 7 однотипных датчиков. Эти числа запоминаются во входном регистре РгВх (цепи синхронизации не показаны) и остаются неизменными на протяжении одного цикла работы устройства. Счетчик в каждом цикле работы устройства вырабатывает последовательность чисел, покрывающую диапазон изменения входных данных. В процессе генерации счетчиком последовательно нарастающих чисел срабатывают компараторы К, для которых число, поступившее со счетчика, оказалось бо́льшим или равным числу, поступившему из входного канала.

Если все 7 чисел различны, то компараторы срабатывают последовательно; если среди этих чисел есть одинаковые, то соответствующие компараторы срабатывают одновременно. Таким образом, по мере отработки элементарного цикла устройства число единиц на выходах компараторов увеличивается. Когда это число достигает заданного порога N или превышает его, срабатывает пороговый элемент ПЭ и на его выходе формируется положительный фронт сигнала М, который вызывает прием кода с выхода счетчика в регистр РгВых.

Код N задает число компараторов, при срабатывании которых в регистре РгВых фиксируется текущее значение последовательно

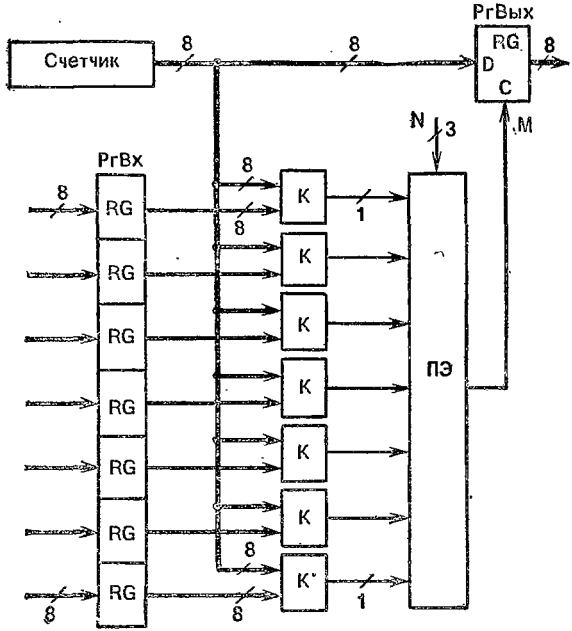


Рис. 8.7. Схема анализа входных сигналов

нарастающего кода на выходах счетчика. При N=1 в регистре РгВых фиксируется минимальное число из поступивших в регистр РгВх, при N=7 — максимальное число, а при промежуточных значениях N — усредненные «взвешенные» числа, в той или иной степени «тяготеющие» к началу или к концу диапазона. При N=4 в регистре РгВых фиксируется значение, которое можно рассматривать как среднее арифметическое множества входных чисел. После срабатывания порогового элемента цикл работы можно завершить и начать снова. При этом в регистр РгВх принимается новая группа чисел, счетчик устанавливается в 0.

8.8. Схемное решение задачи кусочно-линейной аппроксимации нелинейных функций [99]

Пусть требуется быстро вычислить значение одной из функций $y=f_0(x)$, $y=f_1(x)$, ..., $y=f_7(x)$ для заданного значения аргумента x (рис. 8.8, *а*). Задача может быть решена с использованием ПЗУ емкостью 32К×15 бит, как показано на рис. 8.8, *б*. Та же задача решается при значительно меньшей емкости памяти и, возможно, с меньшими потерями времени (все определяется элементной базой), что достигается за счет снижения точности представления функции.

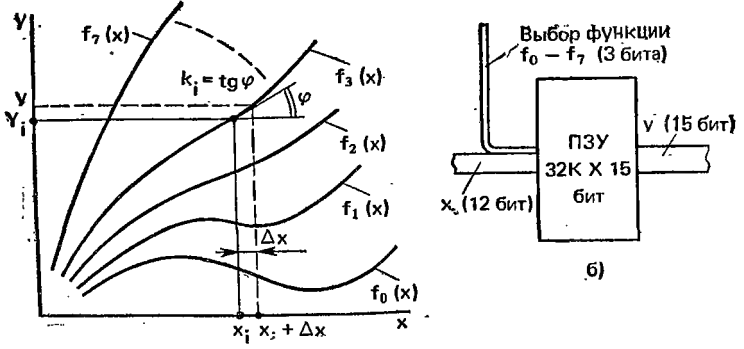


Рис. 8.8. Решение задачи аппроксимации с использованием ПЗУ большой емкости

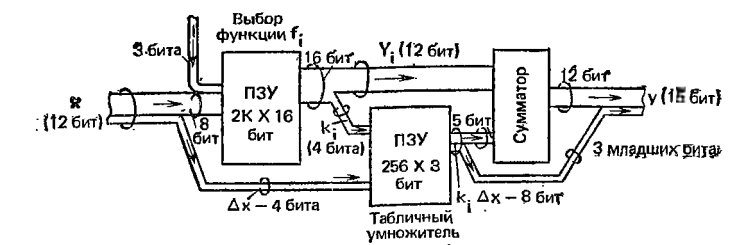


Рис. 8.9. Решение задачи аппроксимации с использованием ПЗУ меньшей емкости

Используется кусочно-линейная аппроксимация. Значение функции вычисляется по формуле $y = Y_I + k_I \Delta x$, обозначения приведены на рис. 8.8, *а*. Схема показаиа на рис. 8.9.

Устройство может быть использовано для ввода данных в микроЭВМ в тех случаях, когда программное вычисление функций недопустимо из-за ограниченной производительности вычислнтельной машины. Возможно также подключение устройства к выходу микроЭВМ для быстрого преобразоваиия выходных данных.

8.9. Быстродействующий генератор кодов маски [100]

При решении многих задач необходимо быстро выделять в слове даиных отдельные биты либо группы бит. В тех случаях, когда программное выделение нужных полей битов неприемлемо из-за больших потерь времени, можно воспользоваться рассмотренными далее схемами.

На рис. 8.10 показан элементарный генератор кодов маски. Код A2 A1 A0 определяет число выходов M7, M6, M5, ..., M0, на ко-

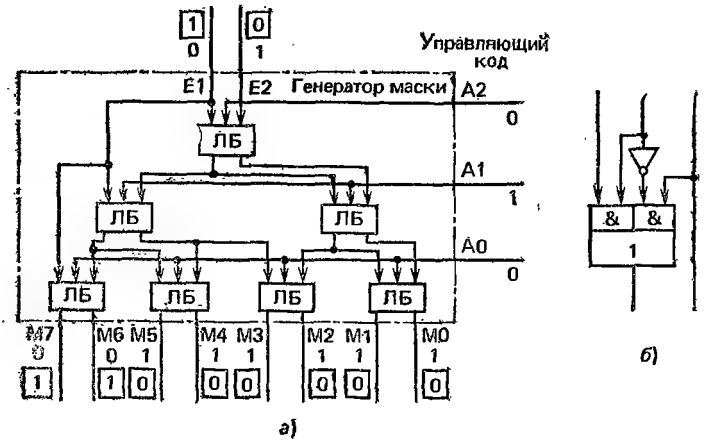


Рис. 8.10. Схема генератора маски *(а)*; структура логического блока ЛБ *(б)*.

Показаны две кодовые комбинации; обведенные и необведенные рамками; A2 — старший разряд управляющего кода

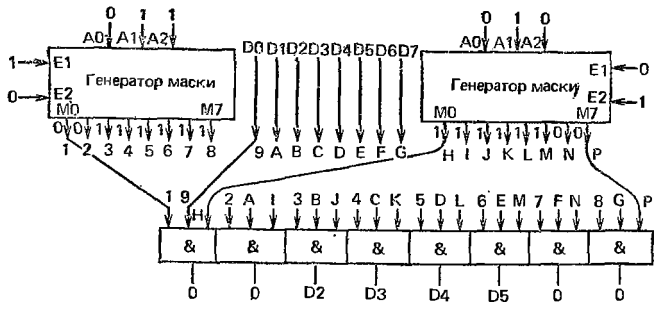


Рис. 8.11. Схема включения двух генераторов маски для двустороннего «обрезания» входного кода D0—D7

торые будут выданы сигналы, совпадающие с сигналом E1. Например, при A2 A1 A0=$010_2=2_{10}$ два выхода M7 и M6 будут повторять значение сигнала E1, как показано на рис. 8.10. Возможен ряд модификаций этой схемы, обладающих повышенным быстродействием. Возможно построение пирамидальных структур на основе генератора для увеличения числа выходов. На рис. 8.11 приведен пример использования генераторов для выделения произвольных (непрерывных) полей бит в 8-разрядном слове данных D0—D7. Из рис. 8.11 видно, что при подаче на вход устройства указан-

ных кодов разряды D0, D1, D6 и D7 входного слова данных теряются и заменяются нулями, а разряды D2—D5 проходят на выход без изменения.

8.10. Устройства для быстрого подсчета числа входных единичных сигналов

Схема устройства [101] приведена на рис. 8.12. По мере наращивания числа единиц на его входах (независимо от их положения) по выходам снизу вверх продвигается «ртутный столбнк» из логических единиц. Схема может служить для сбора и предварительной обработки сигналов с однотипных датчиков, использоваться при построении арбитров и других устройств.

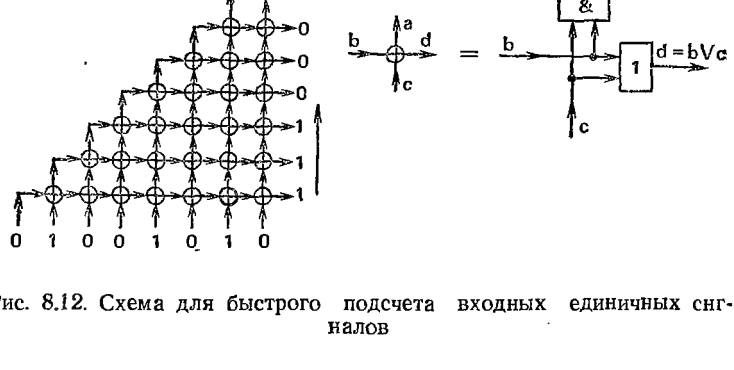


Рис. 8.12. Схема для быстрого подсчета входных единичных сигналов

На рис. 8.13 показана схема порогового логического элемента, построенного с использованием двух одинаковых ПЗУ 256×8 бит с одинаковой кодировкой [144, 178]. Каждое ПЗУ выполняет те же функции, что и схема, приведенная на рнс. 8.12.

При отсутствии единичных сигналов на адресных входах C_i ПЗУ на выходах D0—D7 сформированы сигналы лог. 0. Если на входах ПЗУ присутствует одна единица, то, независимо от ее положения, D0=1, D1=D2=D3=...=D7=0. Прн наличии двух единиц D0=D1=1, D2=D3=D4=...=D7=0. При налнчии трех единиц D0=D1=D2=1, D3=D4=D5=D6=D7=0 и т. д.

В данном примере пороговый логический элемент имеет 17 входов и порог $T=10_{10}$. Единичный сигнал на выходе порогового логического элемента формируется только в тех случаях, когда число единичных сигналов на его входах C1—C17 равно или превышает 10.

8.11. Цифровой фильтр для подавления высокочастотной составляющей сложного сигнала [102]

Под «сложным сигналом» подразумевается сигнал являющийся суммой по меньшей мере двух составляющнх — высоко- и низ-

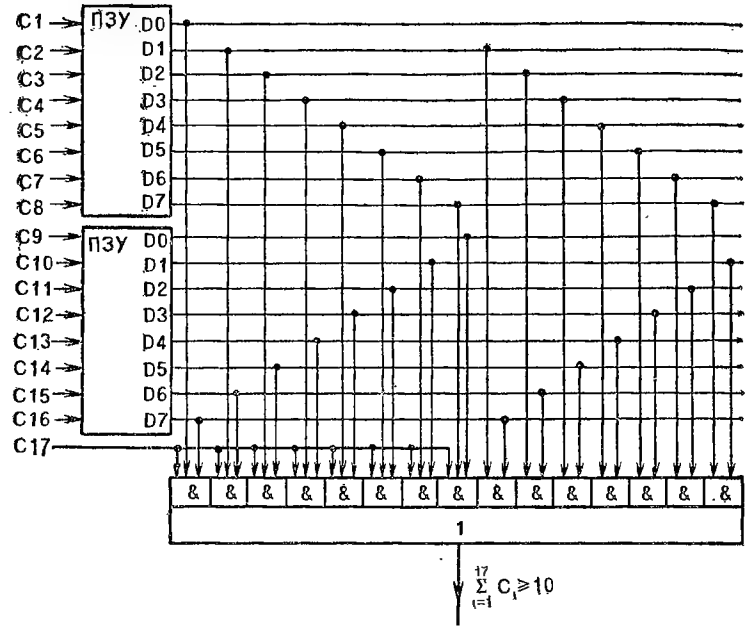


Рис. 8.13. Пороговый элемент на основе двух ПЗУ-«градусников»

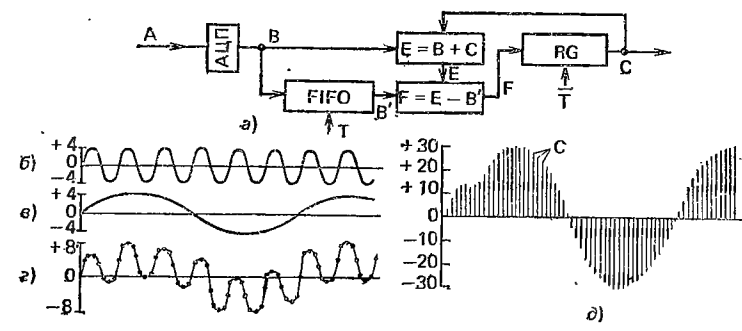


Рис. 8.14. Цифровой фильтр для подавления высокочастотной составляющей сложного сигнала:

а — структурная схема фильтра; *б* — высокочастотная составляющая сигнала; *в* — низкочастотная составляющая сигнала; *г* — суммарный сигнал; *д* — кодированный сигнал на выходе С фильтра

кочастотной. На рис. 8.14, *б — г* показаны соответственно высоко- и низкочастотная составляющие сигнала и их сумма. Задача состоит в подавлении высокочастотной составляющей.

Задача может быть решена аналоговыми средствами с использованием LC- или RC-цепей. Недостаток такого решения — нереализуемость фильтра в монолитном интегральном исполнении и нестабильность его характеристик во времени, при изменении температуры и т. д.

Рассмотрим работу цифрового фильтра (рис. 8.14, *а*).

Фильтр содержит аналого-цифровой преобразователь (АЦП), буфер типа FIFO («первый вошел — первым выходит»), сумматор для арифметического сложения многоразрядных двоичных чисел (со знаком) B и C: E=B+C, вычитатель, выполняющий функцию вычитания F=E—B′, и регистр RG для фиксации текущего значения результата на выходе схемы. Буфер FIFO может быть выполнен, например, в виде 8 последовательно включенных регистров, в каждом из которых временно хранится одно многоразрядное число со знаком. Этот буфер по существу является линией задержки. По каждому положительному фронту синхронизирующего сигнала T происходит продвижение накопленной в буфере информации на одну позицию вправо. Частота синхронизации выбирается такой, чтобы задержка прохождения информации через буфер была равна периоду высокочастотной составляющей сигнала. Выходной регистр стробируется положительными фронтами сигнала $\overline{T}$ для того, чтобы информация на выходах F вычитателя приняла установившееся значение к моменту ее записи в этот регистр. Предполагается, что в исходном состоянии на выходе B АЦП нуль, буфер FIFO пуст, регистр RG устанавлен в 0. Поэтому E=0, B=0, F=0.

При поступлении на вход A устройства сигнала, показанного на рис. 8.14, *г*, на выходе АЦП формируется соответствующая последовательность чисел, которые поступают на сумматор и в буфер. До тех пор, пока неиулевые числа не прошли через буфер, B′=0 и в регистре накапливается алгебраическая сумма чисел, поступающих с выхода АЦП на сумматор. Этот период адаптации соответствует «высокочастотному» выбросу на начальном участке диаграммы, показанной на рис. 8.14, *д*. В дальнейшем, после того как цепочка чисел дошла до выхода B′, вычитатель начинает «взаимно уничтожать» основной и задержаниый высокочастотные сигналы.

8.12. Устройство для инерционной фазовой автоподстройки частоты [15]

При считываиии информации с магнитиого диска необходимо обеспечить синхроиизм между потоком импульсов, поступающих от усилителя считывания, и последовательностью импульсов с генератора. Если генератор «спешит» или «отстает», то иеобходимо воздействовать на него, чтобы ои «подстроился» к потоку считываемых импульсов. Подстройка должна быть плавной, т. е. генератор не должен слишком резко реагировать на управляющие воздействия, корректирующие его работу. Иными словами, генератор должен обладать «инерционностью» (как механический маховик) и в большей степени «доверять» предыстории своей работы, чем отдель-

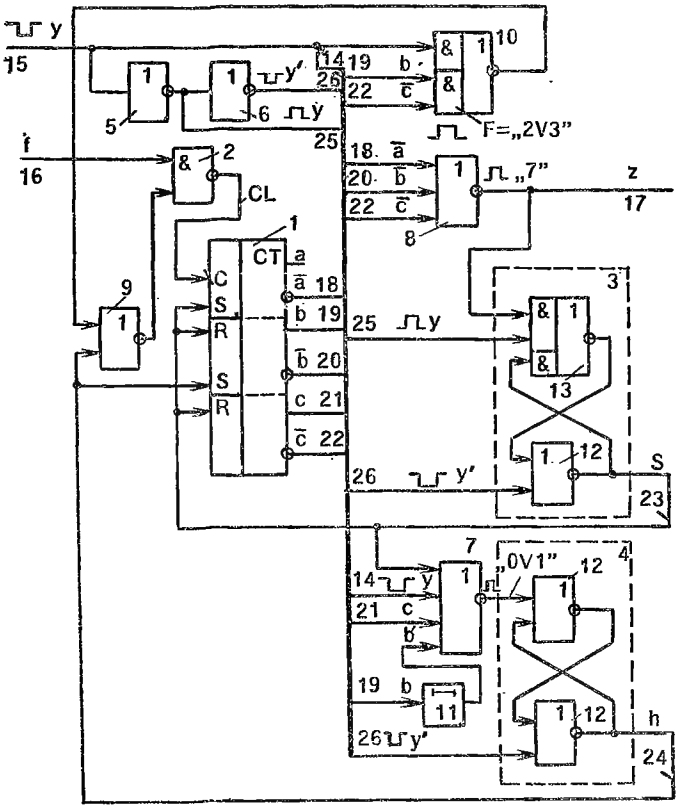


Рис. 8.15. Схема устройства для инерционной фазовой автоподстройки частоты

ным корректирующим воздействиям, хотя и они в своей совокупности должны учитываться.

Это позволяет, во-первых, сохранять правильную синхронизацию при воздействии на вход устройства ложных одиночных импульсов и, во-вторых, противостоять «встречным» фазовым искажениям, при которых соседние информационные импульсы имеют отклонения разного знака относительно идеальных положений.

На рис. 8.15 приведена схема устройства для инерционной фазовой автоподстройки частоты. Устройство содержит счетчик 1, элемент И — НЕ 2, триггеры 3, 4, элементы НЕ 5, 6, элементы ИЛИ — НЕ 7, 8, 9, элемент И — ИЛИ — НЕ 10 и элемент 11 задержки. На вход 16 подается тактовая частота с кварцевого генератора, на вход 15 — импульсы, считанные с диска и «привязанные» к тактовой частоте на входе 16 (схема «привязки» может быть выполнена

на D-триггере, на рисунке он не показан). С выхода 17 устройства снимаются импульсы, имеющие определенное фазовое соотношение с импульсами на входе 15.

Счетчик 1 выполнен по схеме с последовательным переносом и имеет разрядность, равную трем. При наличии сигналов лог. 0 на установочных входах счетчика 1 (точки 23 и 24), т. е. при S= =h=0 счетчик работает в режиме прибавления единицы к старому содержимому по отрицательному фронту сигнала CL на выходе элемента 2, т. е. при переходе этого сигнала из состояния лог. 1 в состоянне лог. 0. Сигналы лог. 1 и лог. 0 представлены соответственно напряжениями высокого и низкого уровней. При переполнении счетчика он вновь начинает счет с нулевого кода (a=b=c= =0).

На рис. 8.16 приведен граф состояний устройства, который отображает восемь возможных состояний счетчика и показывает направления возможных переходов между этими состояниями. Работа счетчика в режиме «естественного» счета (без ускорений и торможеиий) соответствует циклическому перемещеиию по графу от узла к узлу по «большой» окружности в направлении, указанном стрелками.

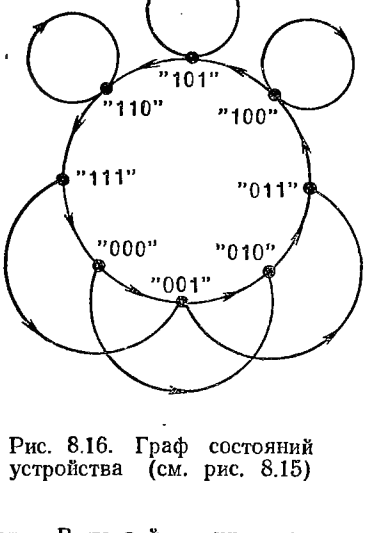


Рис. 8.16. Граф состояний устройства (см. рис. 8.15)

Если S=0, h=1, то средний разряд счетчика принудительно устанавливается в состояние лог. 1 (b=1), а два других разряда остаются без изменения. При работе счетчика в составе устройства возможны только две таиие ситуации. В первой из них счетчик переходит из состояния 000 в состояние 010, во второй — из состояния 001 в состояние 011 (см. соответствующие дуги на рис. 8.16).

Если h=0, S=1, то счетчик принудительно устаиавливается в состояние 001. При работе счетчика в составе устройства возможна единственная ситуация такого рода: переход из сотояния 111 в состояние 001 по внешней дуге графа, приведенного на рис. 8.16. Логика работы устройства такова, что единичные сигналы S и h никогда не формируются одновременно.

При работе счетчика возможны врéменные его приостановки (на один период сигнала f), которые условно показаиы на рис. 8.16 в виде «циклов» (три малые окружности в верхней части рисуика). Такие приостаиовки достигаются формированием сигнала запрета, времеиио запирающего элемент 2 и препятствующего поступлению сигнала CL иа вход синхронизации счетчика.

Элемент задержки 11 компеисирует время задержки распространения сигнала переноса из среднего в старший разряд счетчика 1. Если сигиал переноса имеет максимальную задержку, равную 10 нс, то минимальная задержка элемента 11 может быть вы-

брана равной, например, 15 нс. При этом гарантируется отсутствие ложных срабатываний элемента 7. При использовании счетчика с параллельным переносом условия защиты элемеитов 7, 8 и 10 от ложиых срабатываний при переключении счетчика оказываются иными (здесь не рассматриваются).

Устройство предназначено для выработки на выходе 17 последовательности импульсиых сигиалов z, имеющих определениое фазовое соотношение с импульсиыми сигналами y, поступающими иа

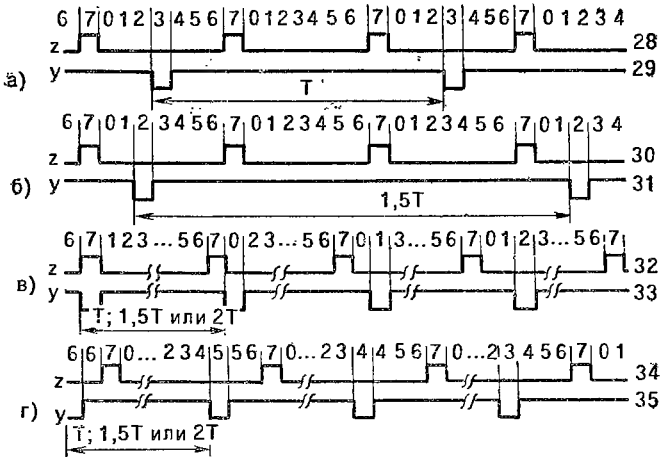


Рис. 8.17. Работа устройства (см. рис. 8.15) без коррекции (*а*, *б*) и с коррекцией (*в*, *г*) состояний счетчика

информационный вход 15 устройства. В идеальных случаях сигналы y должны попадать в середины временных интервалов между сигналами z, как условно показано иа рис. 8.17, *а*, *б* (более детальные временные диаграммы приведены на рис. 8.18). Иными словами, коррекция состояния счетчика не производится, если сигнал y поступил в тот момент, когда в счетчике присутствует код 2 (010) или 3 (011). Сигнал z вырабатывается в те периоды времени, когда в счетчике имеется код 7 (111).

Если сигналы y приходят с упреждением или с опозданием (см. рис. 8.17, *в*, *г*), то счетчик постепенно адаптируется к этим сигналам, ускоряя или замедляя счет путем «перескока» через одно из состояний или приостановки на один такт. В результате сигналы z постепенно приобретают нужное фазовое соотношение с сигналами y.

На рис. 8.17, *в* первый импульс y попадает на код 7 в счетчике, поэтому счетчик «перескакивает» к коду 1, минуя код 0. Это приводит к смещению во времени последовательности сигналов z и поэтому следующий импульс y попадает уже не на код 7, а на

код 0. Счетчик опять реагирует на такую ситуапию «перескоком» к коду 2, миуя код 1 и тем самым вновь смещая времениую диаграмму сигналов z в нужном направлении. Третий импульс у попадает на код 1. Счетчик, минуя состояиие 2, переходит к состоянию 3, продолжая процесс адаптации, и, наконец, четвертый импульс у попадает на код 2 — процесс адаптации завершен. В дальнейшем счетчик работает в «естествением» режиме (без какой-либо коррекции), а импульсы у попадают на код 2. Если импульсы у начинают «отставать» или «обгонять» счетчик, производится соответствующая плавная коррекция его содержимого.

На рис. 8.17, *г* первый импульс «опоздал» (попал на код 6 в счетчике), поэтому счетчик пропускает один такт — в течение двух соседних интервалов времени его содержимое остается равным 6. Благодаря полученному сдвигу времеиной диаграммы сигналов z, следующий импульс опаздывает уже ие так сильно, как первый — он попадает иа код 5, который также не меняется в течеиие двух тактов, следующий импульс у попадает на код 4, счетчик вновь притормаживается и, наконец, четвертый импульс попадает в середину иитервала между импульсами z, что и требовалось.

Период сигнала f, подаваемого на вход 16 сиихронизации устройства, выбирается в 16 раз меньшим, чем миниимальный период T повторения сигналов y (сигналы y в зависимости от считываемой с магнитного диска информации поступают с периодом T, 1,5T или 2T). Рассмотрим работу устройства в восьми возможных ситуациях, по числу возможных кодов в счетчике, на которые «попадает» импульс y.

На рис. 8.18, *а* показана первая ситуация, при которой импульс y поступает на вход 15 устройства в тот момент, когда в счетчике 1 имеется нулевой код (N=0). Согласно рис. 8.16, в этом случае счетчик должеи перейти из состояния 000 в состояние 010 по виешией дуге графа, миуя состояние 001. Этот процесс развивается следующим образом.

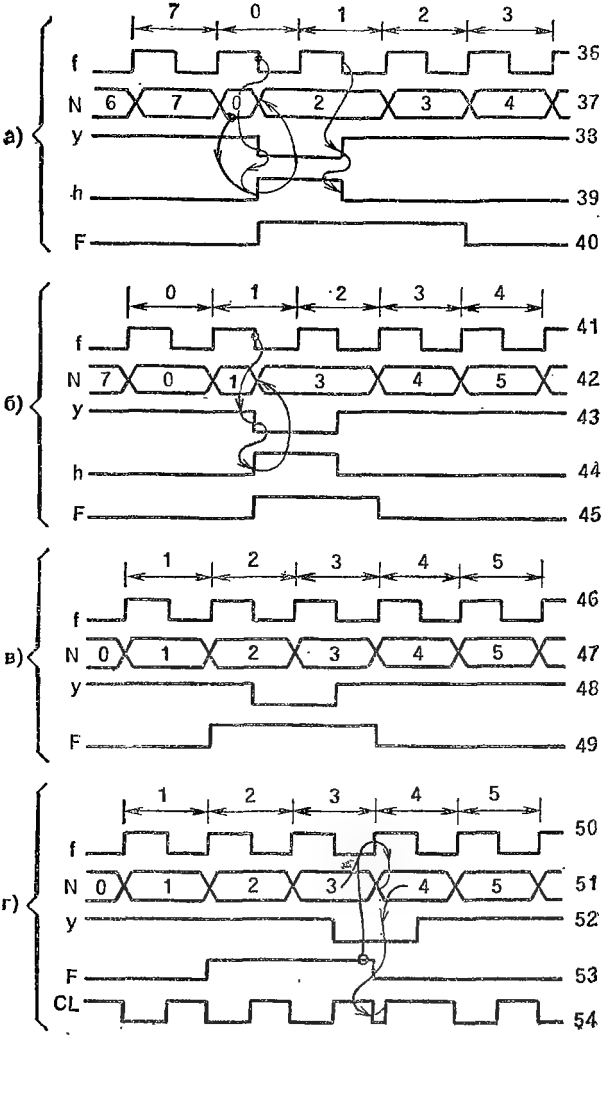
К моменту поступления импульса y h=0, S=0, c=0, b′=0, поэтому импульс y проходнт через элемент 7 и устаиавливает триггер 4 в единичное состояние (h=1), что, в свою очередь, приводит к переходу счетчика в состояние 010. Сигнал h=1 воздействует на вход элемента 9, на его выходе формируется сигнал лог. 0, запрещающий передачу импульса f на вход сиихронизации счетчика. В данном случае выполняется условие F=1.

Рассмотренный процесс заканчивается снятием сигнала y, в результате снимается сигнал h=1 и в дальнейшем счетчик каждый раз прибавляет единицу к своему старому содержимому по положительным фронтам сигнала f. Из диаграмм, приведеиных на рис. 8.18, *а*, видио, что коды N в счетчике меияются не так, как при «естествением» порядке счета (см. верхний ряд цифр: 7, 0, 1, 2, 3).

На рис. 8.18, *б* показана вторая ситуация, при которой импульс y поступает в тот момент, когда в счетчике 1 присутствует код 1. Эта ситуация аналогична предыдущей. Осуществляется переход от кода 1 к коду 3 с пропуском кода 2.

На рис. 8.18, *в* показана ситуация, при которой в момент поступлеиия сигнала y N=2. Благодаря выполнению условия F=1, на выходе элемента 10 формируется лог. 0 и сигнал y ие препятствует прохождению сигнала f на вход CL счетчика. Таким образом, «естественный» счет продолжается.

На рис. 8.18, *г* показана ситуация, при которой в момент по-



ступления сигнала у N=3. К моменту поступления сигнала у F=1, поэтому очередной положительный импульс f вызывает прибавление единицы к содержимому счетчика и в нем устанавливается код 4. При этом начинает выполняться условие F=0 и сигнал y=0, проходя через элементы 10 и 9, вызывает снятие сигнала лог. 0 с входа синхронизации счетчика (см. короткий отрицательный импульс на

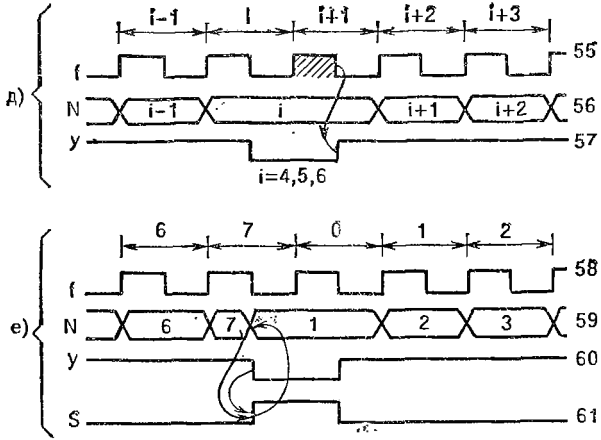


Рис. 8.18. Временные диаграммы работы устройства (рис. 8.15) при попадании входного импульса Y на различные состояния счетчика: *а* — на состояние 0; *б*, *в*, *г* — на состояния 1, 2, 3; *д* — на состояния 4, 5, 6; *е* — на состояние 7

диаграмме 54). После окончания положительного импульса f снимается сигнал y и в дальнейшем счетчик продолжает работу в режиме прибавления единиц к своему содержимому.

Рис. 8.18, *д* поясняет работу устройства в тех ситуациях, когда необходимо пропустить один такт. Эти ситуации соответствуют поступлению сигнала y в те моменты времени, когда в счетчике присутствует код 4, 5 или 6 (см. циклы, показанные на рис. 8.16). Все три ситуации характеризуются выполнением условия F=0, так что сигнал y, проходя через элементы 10 и 9, вызывает запрет прохождения очередного импульса f (на рис. 8.18, *д* этот импульс заштрихован) на вход синхронизации счетчика. Вследствие этого счетчик приостанавливает счет на один такт, что и требуется.

На рис. 8.18, *е* показана ситуация, при которой в момент поступления сигнала y N=7. В данном случае триггер 3 устанавливается в единичное состояние (S=1), причем элемент 7 запирается сигналом S во избежание последующего срабатывания триггера 4 и искажения информации в счетчике. Так как S=1 и h=0, счетчик 1 устанавливается в состояние 001 (см. дугу, соединяющую узлы 111 и 001 на рис. 8.16). По окончании импульса y снимается сигнал S и счетчик продолжает работу в режиме прибавления единиц к своему содержимому в каждом такте.

Устройство входит в состав БИС К1801ВП1-128.

8.13. Устройство для регистрации попадания числа в заданный диапазон [145]

На рис. 8.19 приведена схема, позволяющая определить принадлежность числа X некоторому диапазону чисел, например, заданному так: $L \leqslant X \leqslant H$. Числа L и H предварительно записываются в регистры RG1 и RG2. В регистре RG3, задан код, определяющий вид сравнения: $L \leqslant X < H$, $L < X \leqslant H$, $X = L$ и т. д. При работе схемы число X с помощью компараторов CMP1 и CMP2 одновременно сравнивается с граничными числами L и H, дешифратор DC на основе информации, полученной от компараторов, и в зависимости от

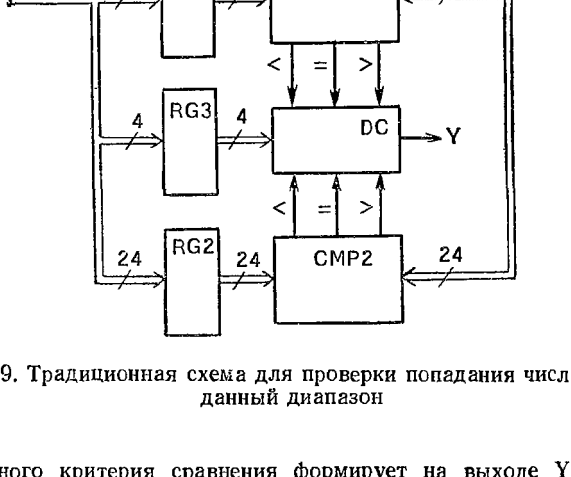


Рис. 8.19. Традиционная схема для проверки попадания числа X в заданный диапазон

выбранного критерия сравнения формирует на выходе Y единичный сигнал, если число X удовлетворяет заданному условию.

Предлагается более простой вариант схемы, показанный на рис. 8.20. Предположим сначала, что разрядность числа X равна 8, т. е. $0 \leqslant X \leqslant FF_{16}$. Старшие 4 разряда числа X подаются на адресные входы ОЗУ1, младшие 4 разряда — на адресные входы ОЗУ2. Выходы a и b ОЗУ1 соединены с управляющими входами A и B мультиплексора MUX (4→1). Выходы c и d ОЗУ2 соединены с информационными входами мультиплексора. При B=A=0 Y=0, так как вход 00 мультиплексора соединен с шиной нулевого потенциала. При B=0, A=1, Y=1, так как на вход 01 мультиплексора подано напряжение высокого уровня. При B=1, A=0 и B=A=1 мультиплексор транслирует на выход Y сигналы с входов 10 и 11.

Предположим, далее, что необходимо формировать сигнал Y=1, если $54 < X < B9_{16}$. Чтобы задать это условие, в ОЗУ1 и ОЗУ2 записывается информация, приведенная на рис. 8.21, *а, б*. Эта информация соответствует разбиению диапазона изменения числа X на 16 страниц, по 16 ячек на странице (рис. 8.21, *в*). В ОЗУ1 «внешние» страницы помечены нулями в разрядах b и a. Иными

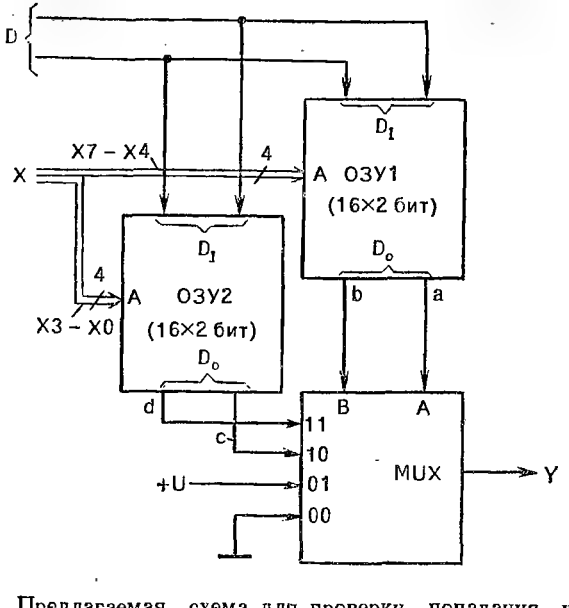


Рис. 8.20. Предлагаемая схема для проверки попадания числа X в заданный диапазон

словами, если старшие 4 разряда кода X (X7—X4) соответствуют цифре в диапазоне 0—4 или C—F, то на управляющие входы A и B мультиплексора поступают нулевые сигналы, следовательно, Y=0.

Аналогично, в ОЗУ1 кодом 01 помечены страницы, полностью лежащие внутри диапазона 54—B9. Поэтому если старшая цифра кода X находится в диапазоне 6—A, мультиплексор формирует сигнал Y=1. В обеих рассмотренных ситуациях (номер страницы соответствует внешним или внутренней областям) сигналы из ОЗУ2 не участвуют в формировании сигнала Y, так как мультиплексор транслирует сигнал либо с входа 00, либо с входа 01 (лог. 0 или лог. 1).

Если старшая цифра числа X принимает значение 5, соответствующее странице, в которой находится нижнее граничное число, то из ОЗУ1 выдается код 10, мультиплексор транслирует на выход Y сигнал с выхода с ОЗУ2. Аналогично, если выбирается страница, в которой размещена верхняя граница диапазона, то на выход Y передается сигнал с выхода d ОЗУ2. Таким образом, ОЗУ1 формирует результат в «очевидных» ситуациях, а ОЗУ2 — в более «тонких», когда необходимо учесть младшие разряды (X3—X0) числа X для проверки выполнения заданного условия.

Запись исходной информации в ОЗУ1 и ОЗУ2 производится с использованием шины D данных и адресной шины обычным путем. Устройство можно запрограммировать так, что диапазон выполне-

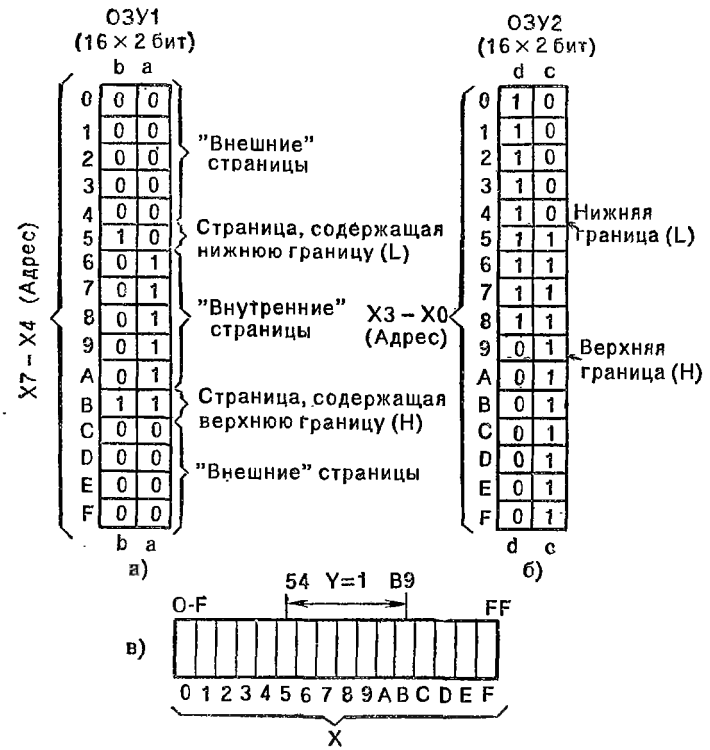


Рис. 8.21. Информация, записанная в ОЗУ1 *(а)* и ОЗУ2 *(б)*, соответствует разбиению диапазона изменения числа X на 16 страниц, по 16 чисел на странице *(в)*

ния условия Y=1 будет не непрерывным, а состоящим из ряда изолированных поддиапазонов; можно проинвертировать условия проверки и т. п.

С увеличением разрядности числа X схема, по существу, не меняется (рис. 8.22, *а*), увеличивается только емкость ОЗУ1 и ОЗУ2, причем не обязательно в равной степени. Если диапазон состоит из одной ячейки, то можно использовать схему, показанную на рис. 8.22, *б*, причем элемент И можно заменить схемой Монтажное И, объединив выходы ОЗУ1 и ОЗУ2, если микросхемы ОЗУ допускают такое объединение.

8.14. Устройство для быстрой сортировки чисел [146]

Устройство, показанное на рис. 8.23, позволяет выбирать максимальное из трех чисел a, b, c: f=max (a, b, c). Устройство содержит три компаратора, логический блок L и мультиплексор

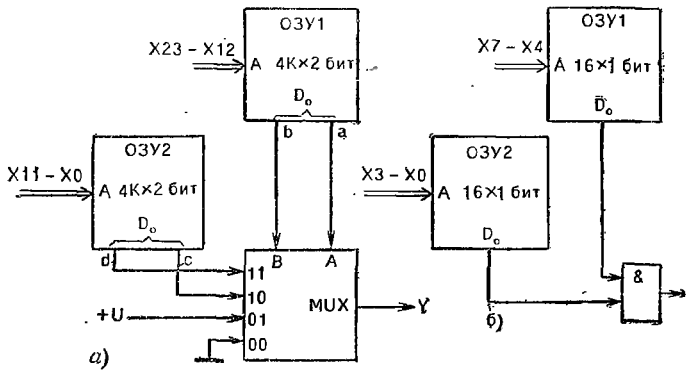


Рис. 8.22. Варианты построения схем:

а — с повышенной разрядностью числа X; *б* — с минимальным диапазоном, соответствующим условию Y=1

MUX. В зависимости от соотношения чисел a, b, c на входы блока L поступают сигналы d, m, n, которые формируются в соответствии с табл. 8.1. Сигналы u, v, w, формируемые блоком L, открывают соответствующие группы усилителей с «третьим состоянием», в результате на выходы устройства передается максимальное число.

Таблица 8.1

Соотношение чисел a, b, c	Сигналы во внутренних точках устройства n	m	d	u	v	w	f
a>b>c	0	0	1	0	1	1	f=a
a>c>b, a>c=b, a=c>b	1	0	1	0	1	1	
a=b=c	1	1	1	0	1	1	
b>c>a	0	1	0	1	0	1	f=b
b>a=c, b>a>c, b=a>c	0	1	1	1	0	1	
c>a>b	1	0	0	1	1	0	f=c
c>b>a, c>b=a, c=b>a	1	1	0	1	1	0	

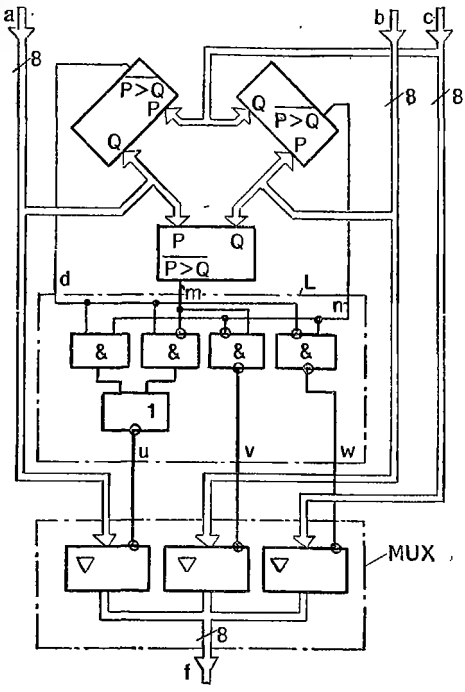


Рис. 8.23. Устройство для выбора максимального числа

Схема, приведенная на рис. 8.24, построена аналогично. Вместо компараторов использованы арифметическо-логические устройства (АЛУ), их выходы Перенос соединены с адресными входами ПЗУ, которое выполняет функции логического блока. Вместо выходов Перенос могут использоваться выходы Знак, вместо ПЗУ можно использовать ПЛМ. Входы r и s используются для выбора нужной функции. Например, устройство может упорядочивать числа a, b, c, d на выходах f1—f4 в порядке возрастания или убывания, при этом можно игнорировать один из входных каналов и т. п. Дополнительные возможности открываются при использовании управляющих входов АЛУ (не показаны).

8.15. Генератор псевдослучайных чисел

Генератор (рис. 8.25) содержит сдвиговый регистр 1, элемент Исключающее ИЛИ 2 и элемент НЕ 3. Выход элемента 2 соединен с информационным входом старшего, 15-го разряда 16-разрядного регистра 1. Входы 4 и 5 регистра 1 являются соответственно входами синхронизации и начальной установки генератора,

а единичные выходы 6 разрядов регистра 1 являются выходами генератора.

Генератор приводится в исходное состояние при воздействии отрицательного (по фазе) импульса на вход 5 начальной установки регистра 1. После окончания импульса в регистре устанавливается некоторый начальный код. Если в качестве входа 5 начальной

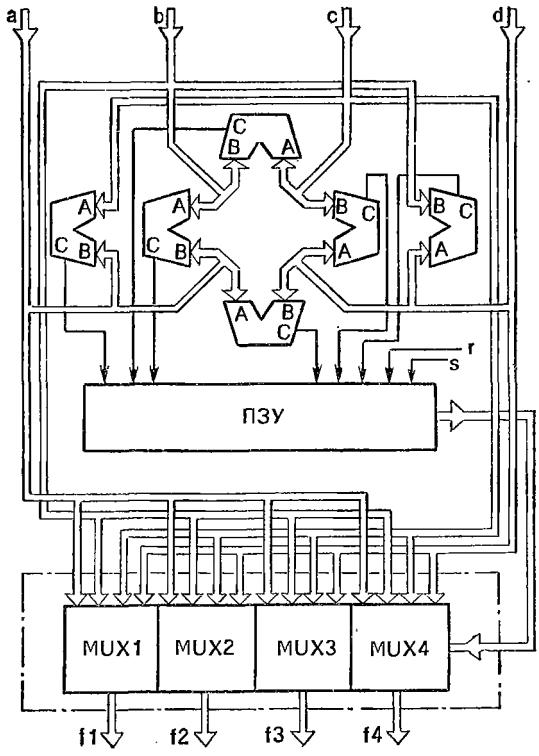


Рис. 8.24. Устройство для быстрой сортировки чисел

установки использован вход R (установка 0), то в регистр записывается нулевой код. Если использован вход S (установка 1), то все разряды регистра переходят в состояние лог. 1. Возможна установка регистра и в другие начальные состояния, за исключением состояний 125252_8 и 052525_8.

В дальнейшем по положительным фронтам импульсов на входе 4 (при переходе сигнала C из состояния лог. 0 в состояние лог. 1) производится сдвиг информации в регистре на один разряд вправо, а в освободившийся разряд (p15) заносится сигнал с выхода элемента 2. Длина цикла (число отличимых друг от друга

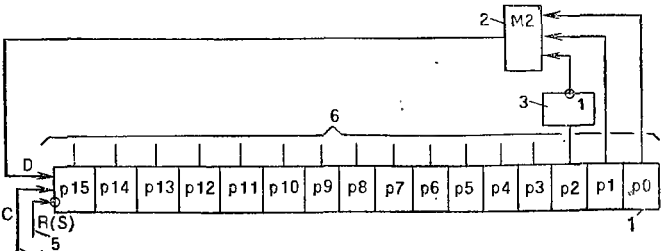


Рис. 8.25. Генератор псевдослучайной последовательности кодов с периодом повторения, равным $2^{16}-2$

кодовых комбинаций) составляет $2^{16}-2$; «недостающие» два кода — 125252_8 и 052525_8.

Схема генератора, приведенная на рис. 8.26, позволяет восполнить «недостающие» коды и получить максимально достижимое число неповторяющихся кодов, равное 2^{16}. Работа генератора поясняется табл. 8.2.

Таблица 8.2

N	Номера разрядов регистра 1																d
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	0	1	0
2	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	0	1
3	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	0
4	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	1
5	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	0

После установки генератора в исходное, например нулевое, состояние (процесс начальной установки совпадает с описанным ранее) он начинает выработку последовательности псевдослучайных кодов при поступлении положительных фронтов сигналов С. Алгоритм формирования кодов тот же, что и реализуемый генератором, показанным на рис. 8.25, так как на начальном этапе работы сигнал d=0 и элемент 2 фактически суммирует три сигнала, как и в предыдущей схеме.

В некоторый момент времени на выходах 6 генератора формируется код, приведенный в первой строке табл. 8.2, а в следующем такте — код, приведенный во второй строке. Дешифратор 7 при этом формирует сигнал d=1, так что в следующем такте (третья строка таблицы) на выходах 6 формируется код 125252, ранее рассматривавшийся как «запрещенный». Так как в данном случае d=0, в следующем такте (четвертая строка таблицы) формируется еще один (оставшийся) «запрещенный» код 052525. Однако «тупиковой» ситуации не возникает, так как сигнал d вновь принимает

единичное значение и в следующем такте (пятая строка таблицы) формируется «правильный» код, который продолжает «естественную» последовательность кодов, прерванную двумя кодами, показанными в третьей и четвертой строках таблицы.

Другими словами, в отсутствие дешифратора или, что то же самое, при использовании схемы, приведенной на рис. 8.25, после кода, показанного во второй строке таблицы, сразу же формируется код, приведенный в пятой строке. Таким образом, в схеме, приведенной на рис. 8.26, коды 125252 и 052525 «вклиниваются» в нормальную последовательность и длина периода неповторяющихся кодов достигает максимального значения, равного 2^{16}.

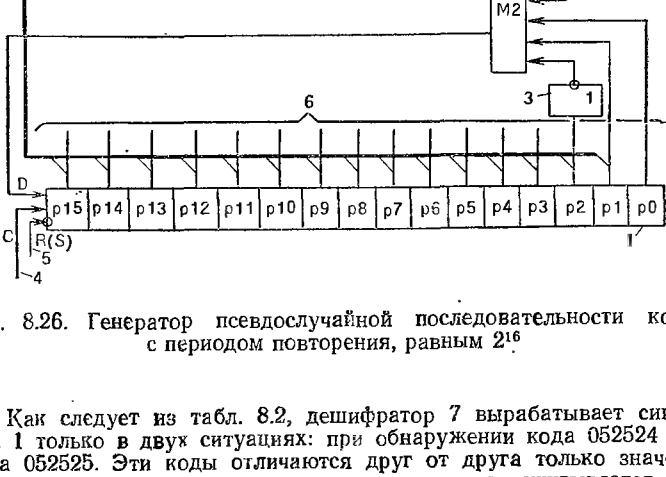


Рис. 8.26. Генератор псевдослучайной последовательности кодов с периодом повторения, равным 2^{16}

Как следует из табл. 8.2, дешифратор 7 вырабатывает сигнал лог. 1 только в двух ситуациях: при обнаружении кода 052524 или кода 052525. Эти коды отличаются друг от друга только значением младшего разряда, поэтому этот разряд не учитывается при дешифрации. Вариант схемы дешифратора показан на рис. 8.27. При наличии сигналов лог. 0 на всех входах элемента ИЛИ — НЕ 11 и сигналов лог. 1 на всех внешних входах элемента И 12 сигнал d принимает единичное значение, что и требуется.

Генератор, показанный на рис. 8.28, по сравнению с генератором, показанным на рис. 8.25, имеет дополнительный вход 10. При подаче отрицательного импульса на этот вход регистр 1 устанавливается в состояние 052525, которое в дальнейшем, с поступлением импульсов на вход 4, постоянно инвертируется: 052525→125252→→052525→125252 и т. д. Этот режим работы генератора полезен при отладке устройств, подключенных к выходам 6 генератора: псевдослучайный процесс временно заменяется строго периодическим, что позволяет обеспечивать устойчивую синхронизацию осциллографа при поиске неисправностей в устройствах.

Подключая элементы НЕ к тем или иным выходам генератора (рис. 8.28), можно вместо указанных кодов 052525 и 125252 получать любые другие пары взаимно обратных кодов. Например, под-

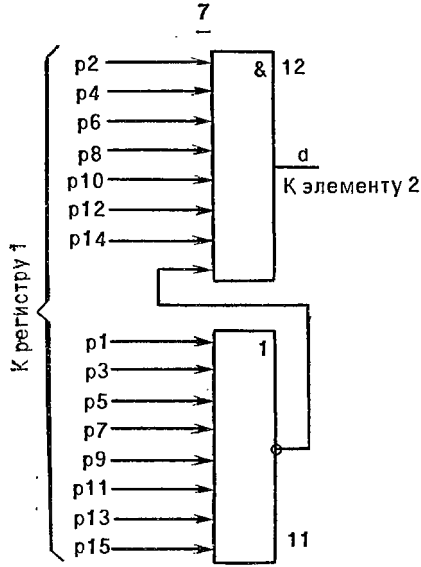


Рис. 8.27. Схема дешифратора 7 (см. рис. 8.26)

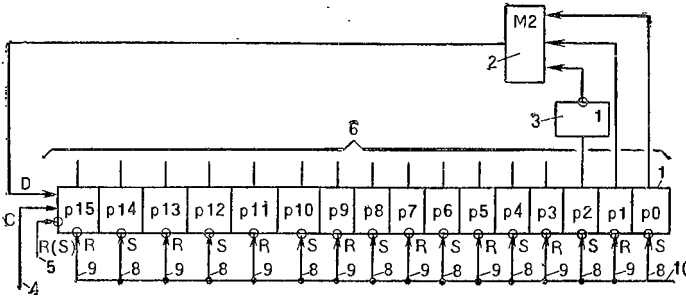


Рис. 8.28. Генератор псевдослучайной последовательности с возможностью работы в режиме периодической выдачи взаимно обратных кодов

ключив элементы НЕ к выходам четных разрядов (один элемент НЕ уже присутствует в схеме), получим два кода: 000000_8 и 177777_8. Применительно к схеме, показанной на рис. 8.25, такое включение элементов НЕ приведет к тому, что «недостающими» кодами будут 000000 и 177777. Аналогично можно получить любую желаемую пару взаимно обратных кодов, кото ые отс тств ют в

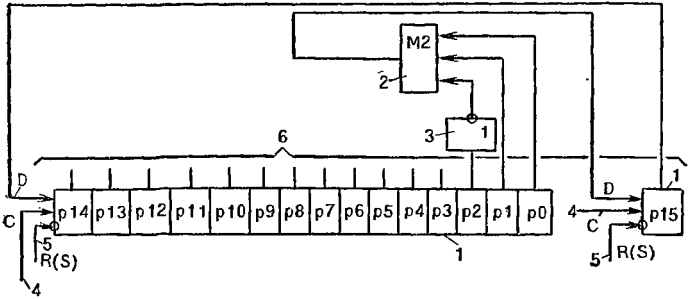


Рис. 8.29. Генератор псевдослучайной последовательности кодов.

Схема аналогична приведенной на рис. 8.25, но в данном случае разряд p14 — старший, а разряд p15 — младший. В программной модели этой схемы разряды p0, p1 и p2 могут использоваться в качестве кода ветвления для перехода к одному из четных адресов следующей команды; при необходимости можно провести перестановку разрядов еще один или несколько раз

последовательности длиной $2^{16}-2$. Вместо элементов НЕ можно использовать двухвходовые элементы Исключающее ИЛИ, первые входы которых подключаются к выходам 6 регистра 1, а вторые — через микропереключатели к шинам лог. 0 и лог. 1. В зависимости от кода, набранного на микропереключателях, будут генерироваться соответствующие пары взаимно обратных кодов.

Отметим, что нумерация разрядов в регистре 1 условная. Любой из разрядов может быть выбран в качестве старшего, младшего или любого из промежуточных. Для получения «быстрой» программной модели генератора удобно разряд p15 «перебросить» в правый конец регистра и считать его младшим разрядом, а разряд p14 — старшим (см. рис. 8.29).

8.16. Аппаратная поддержка операции транспонирования матрицы

На рис. 8.30 показано подключение блока Т транспонирования матрицы данных к магистрали микроЭВМ и структура этого блока. Блок Т содержит 256 триггеров, образующих матрицу 16×16 бит (триггеры показаны точками). При записи адресуемым элементом является строка матрицы; при чтении — столбец. МикроЭВМ построчно загружает в блок Т некоторую информацию из 16 последовательных 16-разрядных ячеек внутреннего ОЗУ, а затем считывает содержимое столбцов, размещая его в тех же последовательных ячейках ОЗУ. В результате производится «поворот» первоначально хранившейся во внутреннем ОЗУ матрицы на 90°. Адресация строк и столбцов блока Т может производиться с использованием стандартных методов дешифрации, см. гл. 1.

Для транспонирования более «мощных» матриц можно использовать ОЗУ с мультиплексором на адресных входах. В примере, приведенном на рис. 8.31, в зависимости от сигнала WR, т. е. от типа обращения (запись или чтение) на адресные входы ОЗУ пе-

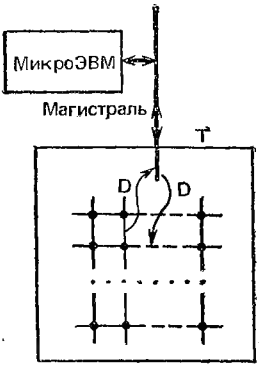


Рис. 8.30. Структура блока транспонирования матрицы данных

редается либо код A7 A6 A5 ... A0 — при записи, либо код A4 A3 A2 A7 A6 A5 A1 A0 — при чтении. (Сигнал VMA подтверждает истинность адреса.) Это позволяет транспонировать матрицу данных, каждый элемент которой состоит из четырех 16-разрядных слов. Каждое слово размещено в одной 16-разрядной ячейке ОЗУ. Элемент матрицы расположен в четырех соседних ячейках ОЗУ и при транспонировании не претерпевает изменений, т. е. рассматривается как неделимый и перемещаемый как единое целое. Размерность матрицы 8×8 элементов. Адреса ячеек ОЗУ связаны с элементами матрицы следующим образом: (см. стр. 499, вверху).

Первый элемент матрицы располагается в четырех ячейках ОЗУ с адресами 000—003$_8$, второй (по строке) — в ячейках с адресами 004—007$_8$ и т. д. При записи по последовательно возрастающим адресам 0, 1, 2, и т. д. мультиплексор MS транслирует сигналы A2—A7 без их перестановки (см. штриховые линии на рис. 8.31), поэтому информация, поступающая по 16-разрядной шине данных D, размещается в ОЗУ в «естественном» порядке и запол-

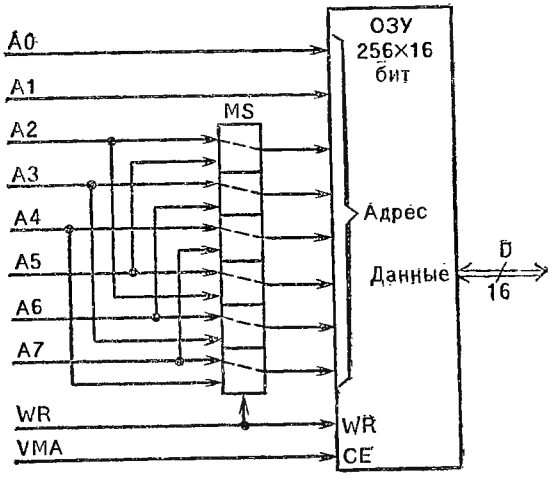


Рис. 8.31. Использование ОЗУ для транспонирования матрицы данных.

Мультиплексор MS управляется сигналом записи WR, поэтому данные записываются по строкам, а считываются по столбцам матрицы

000—003 004—007 010—013 014—017 ... 034—037
040—043 044—047 050—053 054—057 ... 074—077
100—103 104—107 110—113 114—117 ... 134—137
140—143 144—147 150—153 154—157 ... 174—177

340—343 344—347 350—353 354—357 ... 374—377

няет матрицу по строкам слева направо. При считывании также используется адресация последовательных ячеек памяти, однако благодаря перестановке разрядов адреса производится последовательное обращение к ячейкам 000—003, 040—043, 100—103 и т. д., т е. информация считывается по столбцам: сверху вниз, слева направо.

8.17. Быстродействующий компаратор двух чисел [196]

Схемы компараторов двух чисел обычно содержат цепи последовательного переноса признаков сравнения, что снижает их быстродействие. Предлагаемая схема (рис. 8.32) имеет пирамидальную структуру и обеспечивает более высокое быстродействие по сравнению со схемой с последовательным переносом.

Логический блок первого уровня (рис. 8.33, *а*) осуществляет первичную обработку одноименных разрядов A_i и B_i двух операндов A и B (рис. 8.32) в соответствии с табл. 8.3.

Из табл. 8.3 следует, что $c=1$, если $A_i>B_i$, и $d=1$, если $A_i=B_i$. Логические блоки последующих уровней (рис. 8.33, *б*, *в*) выполняют функции в соответствии с табл. 8.4.

Из табл. 8.4 следует, что $i=1$, если соответствующий фрагмент числа A имеет большее численное значение, чем фрагмент числа B; если эти фрагменты одинаковы, то $j=1$.

Таблица 8.3

Входные сигналы		Выходные сигналы	
a	b	c	d
0	0	0	1
0	1	0	0
1	0	1	0
1	1	0	1

Таблица 8.4

Входные сигналы				Выходные сигналы	
e	f	g	h	i	j
0	0	0	0	0	0
0	0	0	1	0	0
0	0	1	0	0	0
0	1	0	0	0	0
0	1	0	1	0	1
0	1	1	0	1	0
1	0	0	0	1	0
1	0	0	1	1	0
1	0	1	0	1	0

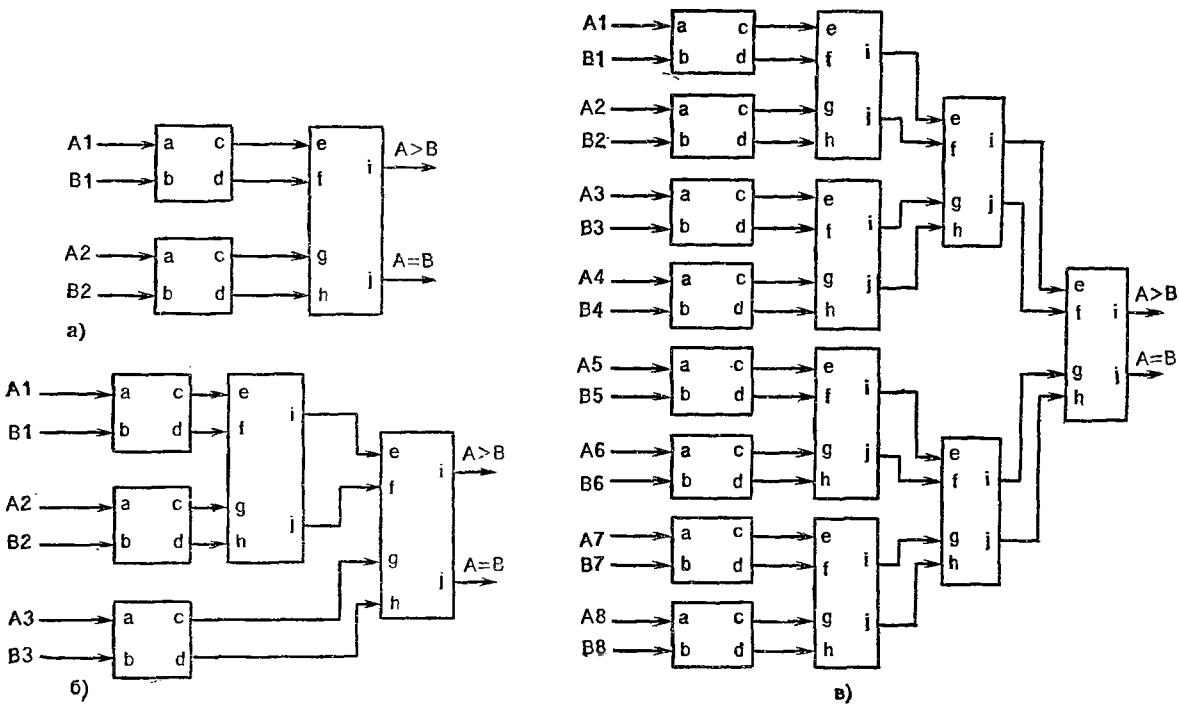


Рис. 8.32. Компараторы для сравнения 2-разрядных (*а*), 3-разрядных (*б*) и 8-разрядных (*в*) чисел.
A1, B1 — старшие разряды

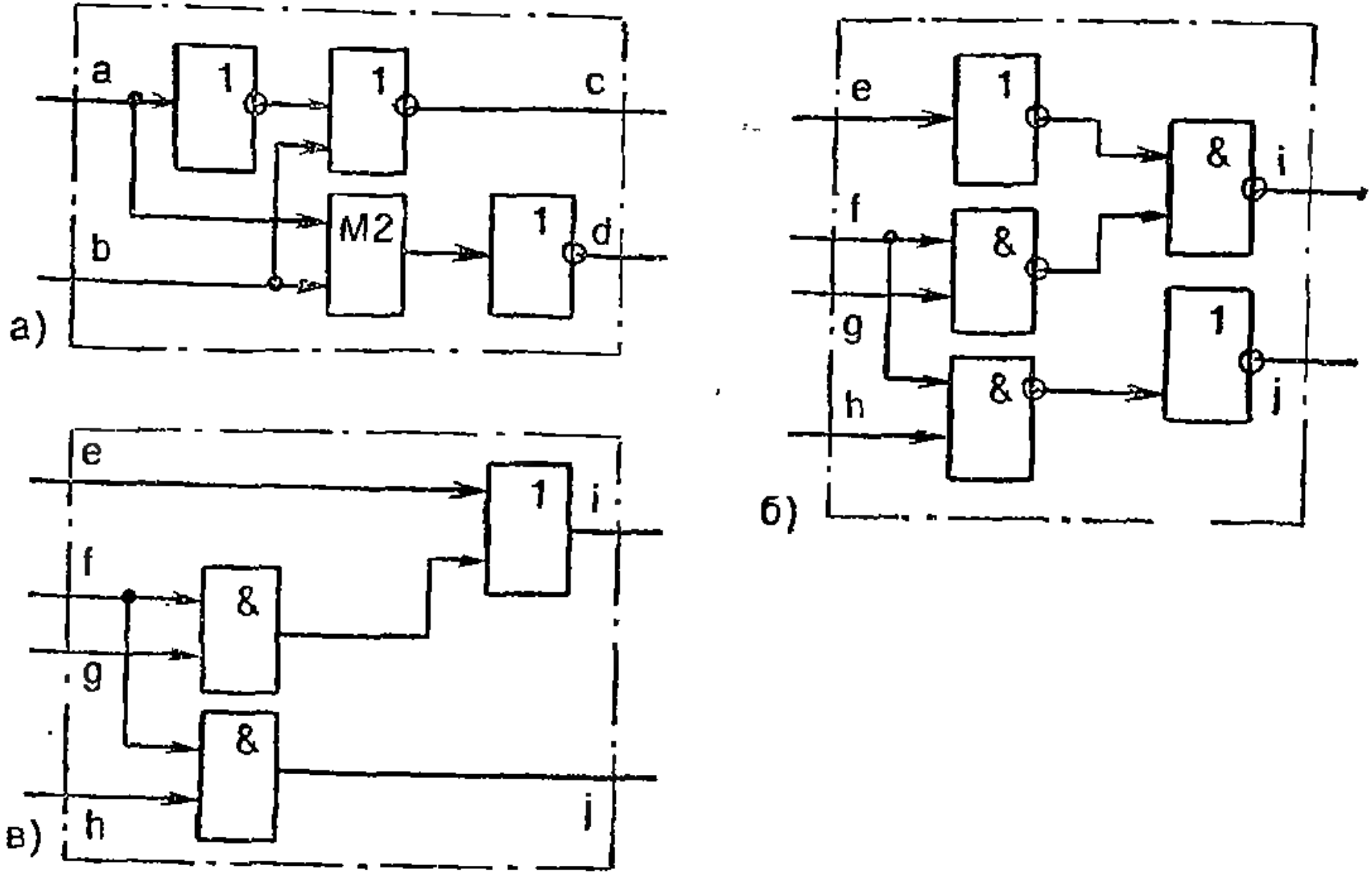


Рис. 8.33. *Логический блок первого (а) и последующих (б, в) уровней; блоки б и в функционально равноценны*

Список литературы

1. Wester J. G., Simpson W. D. Software Design for Microprocessors. — Texas Instr. Learning Center, 1976.
2. Клингман Э. Проектирование микропроцессорных систем: Пер. с англ./Под. ред. С. Д. Пашкеева. — М.: Мир, 1980. — 576 с.
3. PDP Peripherals Handbook. — Digital Equipment Corp., 1975.
4. Microprocessors and Microsystems. — 1979. — Vol. 3, N 6 — R. 284—287; Vol. 3, N 7. — P. 327—336.
5. Пат. 4307448 США.
6. Computer Design. — 1980. — Vol. 19, N 8. — P. 111—118.
7. Пат. 2075225 Великобритания.
8. Пат. 2019057 Великобритания.
9. Electronic Products. — 1982. — Vol. 25, N 5. — P. 63—67, 71—76.
10. А. с. 993267 СССР. — Опубл. в Б. И., 1983, № 4.
11. А. с. 1501075 СССР. — Опубл. в Б. И., 1989. — № 30.
12. А. с. 1515169 СССР. — Опубл. в Б. И., 1989. — № 38.
13. Пат. 4796177 США.
14. А. с. 1566357 СССР. — Опубл. в Б. И., 1990. — № 19.
15. А. с. 1580538 СССР. — Опубл. в Б. И., 1990. — № 27.
16. А. с. 1559352 СССР. — Опубл. в Б. И., 1990. — № 15.
17. Пат. 4223380 США.
18. А. с. 955068 СССР. — Опубл. в Б. И. — 1982. — № 32.
19. А. с. 822185 СССР. — Опубл. в Б. И. — 1981. — № 14.
20. А. с. 1027727 СССР. — Опубл. в Б. И. — 1983. — № 25.
21. Соучек Б. Микропроцессоры и микроЭВМ: Пер. с англ./Под ред. А. И. Петренко. — М.: Сов. радио, 1979. — 520 с.
22. А. с. 746519 СССР. — Опубл. в Б. И., 1980, № 25.
23. Пат. 4177515 США.
24. Пат. 1508854 Великобритания.
25. Пат. 4231021 США.
26. Пат. 4156277 США.
27. Международная заявка PCT/WO/83/01851.
28. Пат. 4271464 США.
29. А. с. 1068926 СССР. — Опубл. в Б. И. — 1984. — № 3.
30. Intern. J. of Electronics. — 1980. — Vol. 49, N 2. — P. 175—177.
31. Computer Design. — 1979. — Vol. 18, N 12. — P. 93—99.
32. EDN. — 1981. — Vol. 26, N 7. — P. 268, 270.
33. The Weekly Journal for Engineers and Managers in Electron. — 1979. — N 164. — P. 13.
34. New Electronics. — 1979. — Vol. 12, № 16. — P. 52.
35. Digital Design. — 1982. — Vol. 12, N 1. — P. 26—35, 68—71.
36. Electronic Packaging and Production. — 1982. Vol. 22, N 5. — P. 155—158.
37. Electronic Design. — 1980. — Vol. 28, N 25. — P. 48 N.
38. Electronic Packaging and Production. — 1981. — Vol. 21, N 4. — P. 115—121.
39. Computer Design. — 1982. — Vol. 21, N 3. — P. 155—162.
40. Electronic Products. — 1983. — Vol. 26, N 4. — P. 75—80.
41. EDN. — 1982. — Vol. 27, N 13. — P. 193.
42. Пат. 2012082 Великобритания.
43. Пат. 4188664 США.
44. А. с. 907550 СССР. — Опубл. в Б. И., 1982, № 7.
45. Пат. 1578820 Великобритания.
46. Пат. 4301504 США.

47. Пат. 4298930 США.
48. New Electronics. — 1981. — Vol. 14, N 10. — P. 28.
49. Пат. 4234872 США.
50. New Electronics. — 1981. — Vol. 14, N 7. — P. 24.
51. Европейский патент ЕР 0071747А1.
52. Пат. 2086625 Великобритания.
53. А. с. 1019451 СССР. — Опубл. в Б. И., 1983, № 19.
54. Пат. 2065939 Великобритания.
55. Пат. 4093985 США.
56. Пат. 4333142 США.
57. Computer Design. — 1981. — Vol. 20, N 12. — P. 187—190.
58. Пат. 2030739 Великобритания.
59. А. с. 945868 СССР. — Опубл. в Б. И. — 1982. — № 27.
60. А. с. 964648 СССР. — Опубл. в Б. И. — 1982. — № 37.
61. А. с. 1100614 СССР. — Опубл. в Б. И. — 1984. — № 24.
62. А. с. 1180912 СССР. — Опубл. в Б. И. — 1985. — № 35.
63. А. с. 1101808 СССР. — Опубл. в Б .И. — 1984. — № 25.
64. А. с. 1159024 СССР. — Опубл. в Б. И. — 1985. — № 20.
65. А. с. 1043621 СССР. — Опубл. в Б. И. — 1983. — № 35.
66. Хоровиц П., Хилл У. Искусство схемотехники: Пер. с англ./Под ред. М. В. Гальперина. Т. 2. — М.: Мир, 1983. — С. 93—97.
67. Европейский патент ЕР 0068099А2.
68. Пат. 4346441 США.
69. Пат. 4318175 США.
70. Пат. 4254463 США.
71. Европейский патент ЕР 0069570А2.
72. Пат. 4302809 США.
73. Пат. 1601955 Великобритания.
74. Пат. 1596523 Великобритания.
75. А. с. 1129615 СССР. — Опубл. в Б. И. — 1984. — № 46.
76. Международная заявка PCT/WO/80/00884.
77. Европейский патент ЕР 0076408А2.
78. Пат. 4241419 США.
79. Пат. 4249240 США.
80. Пат. 4017841 США.
81. Electronic Products. — 1983. — Vol. 25, N 15. — P. 49, 50.
82. New Electronics. — 1981. — Vol. 14, N 23. — P. 23.
83. Европейский патент ЕР 0051332А1.
84. Пат. 1591427 Великобритания.
85. Computer Design. — 1979. — Vol. 18, N 5. — P. 181—189.
86. Пат. 1537419 Великобритания.
87. Пат. 4285036 США.
88. Пат. 4128878 США.
89. Пат. 4031514 США.
90. Пат. 1551450 Великобритания.
91. Пат. 1590871 Великобритания.
92. Пат. 4327407 США.
93. Пат. 1588535 Великобритания.
94. Международная заявка PCT/WO/82/00535.
95. Пат. 2099616 Великобритания.
96. Пат. 2010551 Великобритания.
97. Европейский патент ЕР 0025651А2.
98. Европейский патент ЕР 0075681А2.
99. Пат. 2071947 Великобритания.
100. Пат. 4012722 США.

101. Пат. 1479596 Великобритания.
102. Европейский патент ЕР 0007729В1.
103. EDN. — 1981. — Vol. 26, N 7. — P. 275.
104. Пат. 4156931 США.
105. Пат. 4285038 США.
106. Пат. 2088103 Великобритания.
107. Пат. 1539227 Великобритания.
108. Пат. 2019059 Великобритания.
109. Пат. 1524873 Великобритания.
110. Европейский патент ЕР 0021287.
111. А. с. 1283777 СССР. — Опубл. в Б. И. — 1987. — № 2.
112. А. с. 1229768 СССР. — Опубл. в Б. И. — 1986. — № 17.
113. А. с. 1381528 СССР. — Опубл. в Б. И. — 1988. — № 10.
114. А. с. 1238096 СССР. — Опубл. в Б. И. — 1986. — № 22.
115. Пат. 4635222 США.
116. Пат. 4445175 США.
117. Пат. 4694421 США.
118. Пат. 4517643 США.
119. Пат. 4628449 США.
120. А. с. 1244669 СССР. — Опубл. в Б. И. — 1986. — № 26.
121. Пат. 2013378 Великобритания.
122. Пат. 2047927 Великобритания.
123. Пат. 4584663 США.
124. Electronic Letts. — 1988. — Vol. 24, N 3. — P. 144, 145.
125. Пат. 2175111 Великобритания.
126. Пат. 4616312 США.
127. Пат. 4620277 США.
128. Пат. 4616338 США.
129. Пат. 4654787 США.
130. Пат. 4689740 США.
131. Пат. 4644469 США.
132. Пат. 4638454 США.
133. Пат. 4569045 США.
134. Пат. 4587613 США.
135. Пат. 4675813 США.
136. Пат. 4663728 США.
137. Пат. 4613936 США.
138. Пат. 4635224 США.
139. Пат. 4621318 США.
140. Пат. 4665484 США.
141. Пат. 4652994 США.
142. Европейский патент ЕР 0169909А2.
143. Европейский патент ЕР 0068831А2.
144. Пат. 2152716 Великобритания.
145. Пат. 4682283 США.
146. Пат. 4651301 США.
147. Пат. 4566082 США.
148. Пат. 2125590 Великобритания.
149. Пат. 2122780 Великобритания.
150. Пат. 2181581 Великобритания.
151. Пат. 4096572 США.
152. Европейский патент ЕР 0142983А2.
153. Европейский патент ЕР 0184613А2.
154. А. с. 1410043 СССР. — Опубл. в Б. И., — 1988. — № 26.
155. Пат. 4720810 США.

156. Пат. 4716526 США.
157. Пат. 2177241 Великобритания
158. А. с. 1261107 СССР. — Опубл. в Б. И., 1986, № 36.
159. Международная заявка PCT/WO/84/04983.
160. Пат. 2153562 Великобритания.
161. Пат. 4713759 США.
162. Пат. 2160340 Великобритания.
163. Пат. 2111262 Великобритания.
164. Пат. 2151052 Великобритания
165. Пат. 2144889 Великобритания.
166. Пат. 2159988 Великобритания.
167. Международная заявка PCT/WO/87/06035.
168. Международная заявка PCT/WO/87/06737.
169. Международиая заявка PCT/WO/83/00242.
170. Международная заявка PCT/WO/83/01849.
171. Международная заявка PCT/WO/85/02040.
172. Международная заявка PCT/WO/85/00452.
173. Международная заявка PCT/WO/87/03717.
174. Пат. 4719562 США.
175. Пат. 2121577 Великобритания.
176. Европейский патеит ЕР 0136168А2.
177. Европейский патент ЕР 0177690А2.
178. Европейский патент ЕР 0146632А1.
179. Европейский патент ЕР 0187453А2.
180. Европейский патент ЕР 0080823В1.
181. Европейский патент ЕР 0162653А2.
182. PDP11 Processor Handbook, DEC, 1979, p. 301—303.
183. Европейский патент ЕР 0251056А2.
184. Европейский патент ЕР 0075631В1.
185. Европейский патент ЕР 0256698А2.
186. Европейский патент ЕР 0086601А2.
187. Европейский патент ЕР 0120525А2.
188. Европейский патент ЕР 0192209А1.
189. Европейский патент ЕР 0075684В1.
190. Европейский патент ЕР 0263886А1.
191. Европейский патент ЕР 0044949В1.
192. Пат. 4742452 США.
193. Европейский патент ЕР 0226765А2.
194. Европейский патент ЕР 0242054А2.
195. Европейский патент ЕР 0130429А3.
196. Европейский патент ЕР 0207462А2.
197. Пат. 4751667 США.
198. Европейский патент ЕР 0050919В1.
199. Европейский патент ЕР 0116455А2.
200. А. с. 1434446 СССР. — Опубл. в Б. И. — 1988. — № 40.
201. А. с. 1444785 СССР. — Опубл. в Б. И. — 1988. — № 46.
202. Пат. 4760516 США.
203. Европейский патент ЕР 0142982А2.
204. Пат. 4764896 США.
205. А. с. 1444763 СССР. — Опубл. в Б. И. — 1988. — № 46.
206. А. с. 1462313 СССР. — Опубл. в Б. И. — 1989. — № 8.
207. Пат. 4773040 США.
208. Пат. 4792920 США.
209. Пат. 4803621 США.
210. А. с. 1488820 СССР. — Опубл. в Б. И. — 1989. — № 23.

Оглавление

Предисловие 3

РАЗДЕЛ I

ВВЕДЕНИЕ В ТЕХНИКУ МИКРОЭВМ

Глава 1
Выбор микропроцессорной системы и организация адресного взаимодействия ее составных частей

1.1. С чего начать проектирование микропроцессорной системы? 4
1.2. Типовая структура микроЭВМ, принцип адресного взаимодействия ее составных частей 7
1.3. Методы дешифрации адресов 13
1.4. Схемы адресных дешифраторов 26
1.5. Методы расширения адресного пространства 37

Глава 2
Обеспечение связи микропроцессорной системы с «внешним миром»

2.1. Интерфейс «общая шина» (упрощенная версия) . . 50
2.2. Арбитры, реализующие гибкое обслуживание запросов 63
2.3. Типы контроллеров 68
2.4. Примеры пассивных контроллеров 70
2.5. Методы поиска источников прерываиия 81
2.6. Канал прямого доступа в память 84

Глава 3
Последовательные каналы связи

3.1. Методы кодирования цифровой иифорлации для ее передачи по последовательным каналам связи . . . 88
3.2. Примеры микроЭВМ с магистралями последовательного типа. 91
3.3. Каиалы, использующие код «Манчестер-II» . . . 97
3.4. Каиалы для одновременной двунаправленной передачи информации по одному кабелю 101
3.5. Канал для одновременной передачи иапряжения питания, синхроимпульсов и информации по одному кабелю 105
3.6. Асинхронные двунаправленные приемопередатчики . . 107

Глава 4
Рекомендации по обеспечению помехозащищенности цифровых устройств

4.1. Подавление помех по первичной питающей сети . . . 111
4.2. Правила заземления, обеспечивающие защиту от помех по «земле» 114
4.3. Подавление помех по цепям вторичного электропитания 115
4.4. Правила работы с согласованными линиями связи . . 116
4.5. О защитных свойствах кабелей 122
4.6. Использование оптронных развязок для подавления помех 125

РАЗДЕЛ II

ПРИМЕРЫ ПОСТРОЕНИЯ МИКРОПРОЦЕССОРНЫХ СИСТЕМ И ИХ СОСТАВНЫХ ЧАСТЕЙ

Глава 5
Взаимодействие с объектами

5.1. Опрос удаленного источника информации по двухпроводной линии связи 127
5.2. Схема связи микроЭВМ с мощным потребителем энергии 159
5.3. Передача запросных и ответных сигналов прерывания по двухпроводной линии связи 166
5.4. Передача вектора в процессор, не обладающий способностью реакции на векторные прерывания 167
5.5. Минимизация числа внешних связей устройств при задании их собственных адресов 169
5.6. Определение состояния переключателя без использования схемных элементов опроса 172
5.7. Использование двухвекторных источников прерываний для демократизации их обслуживания 173
5.8. Ввод данных в микроЭВМ с кассетного магнитофона . 177
5.9. Сопряжение микроЭВМ с клавиатурой и со стрелочными индикаторами с использованием общего порта ввода-вывода 178
5.10. Сопряжение микроЭВМ с клавиатурой и с группой датчиков с использованием общего входного порта . . 180
5.11. Подключение клавиатуры к магистрали микроЭВМ . 180
5.12. Подсветка нажатых клавиш ответными сигналами от микроЭВМ без использования дополнительных связей с клавиатурой 181
5.13. Способ идентификации нажатых клавиш 182
5.14. Связь микроЭВМ с удаленным активным датчиком . 183
5.15. Управление отдельными битами выходного порта с использованием адресной шины 184
5.16. Периодическое отключение напряжения питания от энергонезависимых блоков микроЭВМ при выполнении циклических программ 185
5.17. Связь однокристальной микроЭВМ с линейным приемопередатчиком, клавиатурой и цифровым индикатором . 185
5.18. Пример использования однокристальной микроЭВМ в таксометре автомобиля 187

5.19. Использование выходного порта микроЭВМ для ввода информации 189
5.20. Система адресации удаленных устройств, использующая псевдослучайные числовые последовательности . . . 192
5.21. Упрощение конструкции системы электрического согласования кабеля с устройствами — источниками и приемниками информации 196
5.22. Использование сетевого таймера для синхронизации данных, поступающих в микроЭВМ от удаленного источника информации 199
5.23. Использование сетевого таймера для синхронизации обмена данными между двумя микроЭВМ 206
5.24. Система опроса большого числа источников информации 211
5.25. Повышение производительности микроЭВМ при рассасывании очередей из запросов на прерывания . . . 214
5.26. Параллельный опрос ряда источников векторных прерываний (применительно к каналу микроЭВМ типа «Электроника-60») 220
5.27. Об одной особенности подключения программно-доступных элементов к каналу микроЭВМ типа «Электроника-60» 222
5.28. Выявление конфигурации управляемого объекта с использованием диодной схемы ИЛИ 224
5.29. Поиск источника прерывания с использованием группового опроса контроллеров 225
5.30. Клавиатурный блок дистанционного управления с низким потреблением энергии в паузах между сеансами связи 226
5.31. Схема связи микроЭВМ с удаленными абонентами . 229
5.32. Связь микроЭВМ с клавиатурой с использованием цифро-аналогового преобразователя 231
5.33. Уменьшение аппаратных затрат при задании собственных номеров однотипных устройств 233
5.34. Обработка прерываний в многопроцессорной системе . 234
5.35. Программно-управляемое назначение собственных адресов контроллеров 235
5.36. Формирование импульсных сигналов запросов прерываний в потенциальной линии связи типа Монтажное ИЛИ 238

Глава 6
Контроль, диагностика, восстановление, защита и поиск информации

6.1. Метод следящего самоконтроля микроЭВМ на основе предварительного прогнозирования вариантов ее «поведения» 240
6.2. Устройство для контроля работы микроЭВМ путем анализа времени выполнения программного цикла . . . 242
6.3. Программно-управляемая замена одного из блоков памяти на резервный блок 243
6.4. Однокристальная (или одноплатная) микроЭВМ, «создающая себя» при включении напряжения питания из имеющихся исправных блоков 244
6.5. Коррекция постоянных ошибок в ОЗУ методом двойной инверсии 246

6.6. Устройство для обнаружения ошибок — двух последовательных записей в одиу и ту же ячейку ОЗУ . . . 247
6.7. Устройство для оперативного контроля магистрали микроЭВМ, использующее паузы в ее работе 248
6.8. Оперативный коитроль магистрали микроЭВМ с возможностью адаптации к отказам информационных и адресных линий 249
6.9. Исправление постояиной ошибки в линии магистрали за счет «логической поддержки» ближайших соседних линий 250
6.10. Коррекция одиночных отказов информационно-адресной шины путем инвертирования передаваемых кодов . . 252
6.11. Подключение абонентов к адресной шине с возможностью коррекции ошибок в ее линиях 252
6.12. Устройство для контроля управляющей вычислительной машины 255
6.13. Передача управления при «зависаниях» путем доопределения магистральных сигналов 258
6.14. Контроль сохранности данных в статическом ОЗУ с резервным источником питания 259
6.15. Устройство для коррекции ошибок при передаче или хранении информации 271
6.16. Защита программы от ее прогона на «чужой» микроЭВМ, однотипной со «своей» 274
6.17. Перемещение отказавших страниц динамического ОЗУ в конечную его часть 277
6.18. Резервированная система из трех микроЭВМ . . . 281
6.19. Точный или приближенный поиск желаемой последовательности символов в проходящем потоке символов . 283
6.20. Поиск в ОЗУ слова, минимально отличающегося от эталонного слова 287
6.21. Устройство для защиты информации в коллективной памяти от посторонних пользователей 288
6.22. Повышение обнаруживающей способности системы контроля, использующей код Хэмминга 289
6.23. Повышение вероятности обнаружения ошибок путем их размножения 291
6.24. Схема защиты информации в ОЗУ от ошибочной записи 296
6.25. Контроль выполнения программы с использованием таймера 298
6.26. Контроль передачи адресных сигналов к микросхемам памяти 299
6.27. Контроль передачи сигналов из микроЭВМ к управляемому объекту 301
6.28. Коитроль данных по четности с повышенной вероятностью обнаружения групповых ошибок 304
6.29. Контроль информации, считываемой из ОЗУ, без снижения его быстродействия 306
6.30. Восстановление информации, поступающей в микроЭВМ из нескольких каналов связи 307
6.31. Восстановление информации при ее приеме из параллельного канала связи 312
6.32. Схема регистрации устойчивых одиночных ошибок в динамическом ОЗУ 314

6.33. Схема слежения за адресами команд, выполняемых процессором 315
6.34. Байтовый канал связи с чередованием во времени проверок нечетности и четности числа передаваемых единиц 317

Глава 7
Адресация, обмен данными

7.1. О преимуществах псевдослучайной адресации памяти . 319
7.2. Размещение оперативной памяти данных микроЭВМ вне ее адресного пространства 320
7.3. Расширение адресного пространства микроЭВМ при плотном его заполнении «окнами» 321
7.4. Преобразователь логических адресов в физические, обеспечивающий конвейерный режим работы блоков памяти 322
7.5. Повышение производительности микроЭВМ за счет опережающего выбора команд и данных из памяти . . 323
7.6. Гибкое использование абсолютных адресов, хранимых в программном ПЗУ 326
7.7. Улучшенная структура буфера типа FIFO 328
7.8. Оперативная память с возможностью одновременного чтения нескольких ячеек 331
7.9. Определение истинности передаваемой по магистрали информации без использования сопровождающего синхросигнала 332
7.10. Автоматическое назначение собственных адресов блоков памяти микроЭВМ при включении напряжения питания 333
7.11. Помехоустойчивая магистраль типа Монтажное ИЛИ с использованием дифференциальных пар 338
7.12. Арбитр распределенного типа 338
7.13. Четырехпроводная магистраль для двунаправленной передачи информации в вычислительной системе . . . 339
7.14. Перемещение свободных областей памяти в конечную ее часть без приостановки выполнения рабочих программ 343
7.15. Сокращение числа выводов однокристальной микроЭВМ за счет преобразования адресных сигналов 345
7.16. Уменьшение числа линий адресной шины при подключении ОЗУ к процессору 346
7.17. Компактное размещение пользовательских адресных пространств в общей физической памяти 349
7.18. Схемное определение наиболее «устаревшего» блока буферной памяти 351
7.19. Варианты построения арбитров, реализующих гибкое обслуживание абонентов 352
7.20. Устройство для быстрого стирания информации в памяти микроЭВМ 366
7.21. Использование освободившихся при частичной дешифрации разрядов адресной шины для передачи данных . 371
7.22. Трехпроводная магистраль для передачи переменного напряжения питания, информации и стартовых меток . 372
7.23. Микропроцессорная управляющая система с битовой и байтовой адресацией ОЗУ 375
7.24. Использование страничного режима работы микросхем динамической памяти для повышения производительности мнкроЭВМ 377

7.25. Управление программным семафором без использования режима Чтение — модификация — запись 395
7.26. Использование динамического ОЗУ для хранения констант, «не разрушающихся» при выключении напряжения питания 397
7.27. МикроЭВМ с программируемыми адресами контроллеров 404
7.28. Ускорение записи неполноразрядных слов в память с большой разрядностью шины данных 405
7.29. Схема формирования ответных сигналов от абоентов, учитывающая различие их быстродействия . . . 408
7.30. Уменьшение потерь производительности микроЭВМ, связанных с регенерацией динамической памяти . . . 409
7.31. Управление программными семафорами в многопроцессорной системе 412
7.32. Повышение производительности двухмашинной системы с общим блоком памяти 416
7.33. Изменение параметров устройства, задаваемых микропереключателями, в отсутствие доступа к этим микропереключателям 417
7.34. Подключение памяти на магнитных дисках к многопроцессорной системе 419
7.35. Обмен информацией между процессорами через буфер типа FIFO 421
7.36. Многопроцессорная система, использующая табличную память для выбора страницы в главной памяти . . 422
7.37. Уменьшение аппаратуры дополнительной памяти микроЭВМ 422
7.38. Коррекция информации, хранимой в ПЗУ, с помощью дополнительной аппаратуры 424
7.39. Простой интерфейс с совмещенной шиной адресов — данных 430
7.40. Персональная микроЭВМ с возможностью использования процессоров разных типов 432
7.41. ПЗУ со страничной организацией 433
7.42. Расширение памяти микроЭВМ подключением дополнительных блоков разного объема 437
7.43. Программируемый адресный дешифратор 441
7.44. Взаимодействие между центральной и периферийными микроЭВМ с использованием общего ОЗУ 442
7.45. Повышение быстродействия кэш-памяти 446
7.46. Магистраль с электрическими характеристиками, не зависящими от числа подключенных к ней устройств . 451
7.47. Арбитр двухпортового ОЗУ 452
7.48. Конструкция магистрали, допускающая ее разделение на изолированные друг от друга части 455
7.49. Быстродействующий распределенный арбитр . . . 457
7.50. Ускорение программно-управляемой пересылки данных при использовании операции фиктивного чтения . . 460
7.51. Подтверждение истинности передаваемых данных попеременно фронтом и срезом синхросигнала 465

Глава 8
Аппаратиая поддержка вычислений, цифровые фильтры, узлы микроЭВМ

8.1. Расширеиие системы комаид процессора путем подключения к нему программно-доступиых элементов . . 469
8.2. Распараллеливание выбора двух операндов при выполнении двухадресных команд 470
8.3. Аппаратная поддержка механизма вызова подпрограмм 470
8.4. Аппаратная реализация вызова произвольных участков программ, не являющихси «оформленными» подпрограммами 472
8.5. Микропрограммиая поддержка механизма выбора из памяти «разбросанных» данных 473
8.6. Цифровой фильтр для сглаживаиия ступенчатых воздействий на механические исполнительные органы . . 474
8.7. Цифровой фильтр для усреднения зиачений входных сигналов 475
8.8. Схемное решение задачн кусочно-линейной аппроксимации нелинейиых функций 476
8.9. Быстродействующий генератор кодов маски . . . 477
8.10. Устройства для быстрого подсчета числа входных единичных сигиалов 479
8.11. Цифровой фильтр для подавления высокочастотиой составляющей сложного сигнала 479
8.12. Устройство для инерционной фазовой автоподстройки частоты 481
8.13. Устройство для регистрации попадания числа в заданный диапазон 488
8.14. Устройство для быстрой сортировки чисел 490
8.15. Генератор псевдослучайных чисел 492
8.16. Аппаратная поддержка операции транспонирования матрицы 497
8.17. Быстродействующий компаратор двух чисел . . . 499
Список литературы 502